

Universidade Federal de Uberlândia
Faculdade de Computação
Programa de Pós-Graduação em Ciência da Computação



**NOVAS ABORDAGENS NA EVOLUÇÃO DE AUTÔMATOS
CELULARES APLICADOS AO ESCALONAMENTO DE
TAREFAS EM MULTIPROCESSADORES**

Paulo Moisés Vidica

Uberlândia - MG

Janeiro de 2007

Dados Internacionais de Catalogação na Publicação (CIP)

V653n	Vidica, Paulo Moisés, 1976- Novas abordagens na evolução de autômatos celulares aplicados ao escalonamento de tarefas em multiprocessadores / Paulo Moisés Vidica - 2007. 236 f. : il. Orientadora: Gina Maira Barbosa de Oliveira. Dissertação (mestrado) - Universidade Federal de Uberlândia, Programa de Pós-Graduação em Ciência da Computação. Inclui bibliografia. 1. Inteligência Artificial – Teses. 2. Algoritmos genéticos – Teses. I. Oliveira, Gina Maira Barbosa de. II. Universidade Federal de Uberlândia. Programa de Pós-Graduação em Ciência da Computação. III. Título. CDU: 681.3:007.52
-------	--

Elaborado pelo Sistema de Bibliotecas da UFU / Setor de Catalogação e Classificação

NOVAS ABORDAGENS NA EVOLUÇÃO DE AUTÔMATOS CELULARES APLICADOS AO ESCALONAMENTO DE TAREFAS EM MULTIPROCESSADORES

Por

Paulo Moisés Vidica

DISSERTAÇÃO APRESENTADA À
UNIVERSIDADE FEDERAL DE UBERLÂNDIA,
MINAS GERAIS, COMO PARTE DOS REQUIS-
ITOS EXIGIDOS PARA OBTENÇÃO DO TÍTULO
DE MESTRE EM CIÊNCIA DA COMPUTAÇÃO

Área de concentração: Inteligência Artificial.

Orientadora: Profa. Dra. Gina Maira Barbosa de
Oliveira

JANEIRO DE 2007

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE COMPUTAÇÃO

Os abaixo assinados, por meio deste, certificam que leram e recomendam para a Faculdade de Computação a aceitação da dissertação intitulada “**Novas Abordagens na Evolução de Autômatos Celulares Aplicados ao Escalonamento de Tarefas em Multiprocessadores**” por **Paulo Moisés Vidica** como parte dos requisitos exigidos para a obtenção do título de **Mestre em Ciência da Computação**.

Uberlândia, Janeiro de 2007

Orientadora:

Profa. Dra. Gina Maira Barbosa de Oliveira
Universidade Federal de Uberlândia UFU / MG

Banca Examinadora:

Prof. Dr. Keiji Yamanaka
Universidade Federal de Uberlândia UFU / MG

Prof. Dr. André Ponce de Leon F. de Carvalho
Universidade de São Paulo USP / SP

Data: Janeiro, 2007

Autor : **Paulo Moisés Vidica**
Título: **Novas Abordagens na Evolução de Autômatos Celulares Aplicados
ao Escalonamento de Tarefas em Multiprocessadores**
Faculdade: **Faculdade de Computação**
Grau: **Mestrado**

Fica garantido à Universidade Federal de Uberlândia o direito de circulação e impressão de cópias deste documento para propósitos exclusivamente acadêmicos, desde que o autor seja devidamente informado.

Autor

O AUTOR RESERVA PARA SI QUALQUER OUTRO DIREITO DE PUBLICAÇÃO DESTE DOCUMENTO, NÃO PODENDO O MESMO SER IMPRESSO OU REPRODUZIDO, SEJA NA TOTALIDADE OU EM PARTES, SEM A PERMISSÃO ESCRITA DO AUTOR.

Dedicatória

À minha esposa Kédma Campos, aos meus pais Darcil e Maria, à minha irmã Lorena Cristina, ao meu filho Gabriel Campos Vídica e à minha sogra Alice Fonseca de Campos.

Agradecimentos

Primeiramente, agradeço a Deus, acima de tudo, por todas as oportunidades que tem me dado e por tudo o que tem feito em minha vida.

À Profa. Dra. Gina Maira Barbosa de Oliveira, uma orientadora brilhante, sempre muito atenciosa, paciente e compreensiva que, em todos os momentos, deu-me a orientação necessária para a realização das pesquisas e trabalhos. Sou muito grato por tudo e tenho por ela imensa admiração.

À minha esposa e companheira Kédma Campos, que foi, sem dúvida, minha maior incentivadora na busca de mais uma conquista em minha vida, o título de Mestre em Ciência da Computação. Sempre muito paciente e compreensiva, soube entender o quão é importante, para nós, a conclusão deste trabalho. Deu-me força e muito apoio nos momentos mais difíceis. Meu amor, muito obrigado.

Aos meus pais Darcil da Costa Vidica e Maria Flomira da Costa, e à minha irmã Lorena Cristina Vidica, que representam a base de tudo o que conquistei, porque eles me educaram permitindo enfrentar, de cabeça erguida, todos os desafios impostos pela vida.

Aos diretores da SWB (Software Brasil Soluções Integradas), empresa onde trabalho há 6 anos, por acreditarem no meu potencial e permitirem que eu me ausentasse, diversas vezes, do trabalho para me dedicar aos trabalhos relacionados ao Mestrado. Abdalla Hanna Atux Neto, José Maria Lobato e Cintia Pereira Rocha, muito obrigado.

À toda a minha família, em especial aos meus primos Alex Menezes Vidica e Sérgio de Paula Rosa, por todo apoio que me foi dado durante os trabalhos, especialmente por me emprestarem seus computadores, para execução dos trabalhos, em algumas idas a Itumbiara-GO.

Aos amigos Luciene Chagas de Oliveira, Elnor Ribeiro, Rodrigo Vasconcelos Salvo, Douglas Re-

sende Maciel, Kairon, Fabiano Silvério, Lucas Bucci da Silveira e Henrique Costa Neto, pelo incentivo, pelas discussões teóricas e sugestões, e pela ajuda na realização de alguns trabalhos.

Aos membros da banca, Prof. Dr. Keiji Yamanaka e Prof. Dr. André Ponce de Leon F. de Carvalho pela colaboração.

À todos os professores e amigos da Pós Graduação da Universidade Federal de Uberlândia, sou grato pelo compartilhamento de conhecimentos, pelo companheirismo e pela amizade.

À minha sogra Alice Fonseca de Campos que, infelizmente, deixou-nos poucas horas antes de minha defesa. Ela foi uma mulher maravilhosa, muito dedicada ao marido, filhos, netos, enfim, a todas as pessoas próximas a ela. Ela foi uma mãe para mim e sempre torceu pelo meu sucesso. Dona Alice, muito obrigado e descanse em paz, ao lado de Deus.

Enfim, deixo aqui o meu muito obrigado a todos que contribuíram, de alguma forma, para a realização de mais um de meus sonhos.

Resumo

O escalonamento de tarefas em uma arquitetura multiprocessadora é ainda um grande desafio na área de computação paralela. Neste trabalho, estudamos um algoritmo de escalonamento baseado em autômatos celulares (ACs) que tem o objetivo de alocar tarefas de um programa paralelo em um sistema com dois processadores. O algoritmo de escalonamento apresenta duas fases: a fase de aprendizagem e a fase de operação. O propósito da fase de aprendizagem é descobrir regras de ACs aptas ao escalonamento das tarefas. A busca por estas regras é conduzida com a utilização de um algoritmo genético (AG). Na fase de operação, as regras descobertas na fase anterior são aplicadas em novas instâncias de programas paralelos. É esperado que, para qualquer alocação inicial das tarefas, o AC seja apto a encontrar uma alocação onde o tempo total de execução T seja minimizado, ou muito próximo disso. Estudamos inicialmente os modelos de ACs e AGs propostos e publicados até então para a arquitetura do escalonador de tarefas. Após o entendimento e reprodução de alguns resultados publicados, a meta do trabalho passou a ser investigar a capacidade de generalização das regras de transição de ACs. Ou seja, investigar se as regras encontradas para um programa paralelo específico poderiam ser aplicadas, com sucesso, em outros programas. A principal conclusão dessa investigação é que ainda existe muito espaço para a melhoria dessa capacidade. Visando melhorá-la, apresentamos duas novas abordagens para a fase de aprendizagem do algoritmo de escalonamento baseado em ACs: a evolução conjunta e um ambiente coevolutivo. Resultados obtidos através destas novas abordagens mostram que, com o seu uso, as regras de ACs evoluídas apresentam uma melhor capacidade de generalização.

Palavras-chave: Autômatos Celulares, Algoritmos Genéticos, Escalonamento de Tarefas em uma Arquitetura Multiprocessadora

Abstract

Scheduling tasks in multiprocessor architectures still is a challenge in parallel computing field. In this work, we studied a scheduling algorithm based on cellular automata (CA) with the goal of allocate parallel program tasks in a system with two processors. The scheduling algorithm has two phases: a learning phase and an operating phase. The purpose of the learning phase is to discover CA rules for scheduling. A genetic algorithm (GA) is used for search these rules. In the operating phase, the rules discovered in the previous phase are applied in new instances of parallel programs. It is expected that for any initial allocation of the tasks, CA will be able to find an allocation of tasks where the total execution time T is minimized (or close to it). We first studied CA and GA models proposed and published for the task scheduler architecture. After the understanding of these models and the reproduction of some published results, our goal turned to study the generalization ability of the CA transition rules. We investigated if the rules found for a specific parallel program can be applied, successfully, in other programs. Our main conclusion about this investigation is that there is a lot of space for improving this ability. Aiming to improve this generalization ability, we present two new approaches for the learning phase of the scheduling algorithm based on CA: the joint evolution and a coevolutionary environment. Results obtained through these new approaches show that, applying them, the evolved CA rules present a better generalization ability.

Keywords: Cellular Automata, Genetic Algorithms, Scheduling Tasks in Multiprocessor Architectures

Sumário

Lista de Figuras	xvii
Lista de Tabelas	xxi
Glossário	xxix
1 Introdução	1
1.1 Objetivo	3
1.2 Justificativa	3
1.3 Contribuições	4
1.4 Estrutura do Trabalho	5
2 Algoritmos Genéticos	7
2.1 Inteligência Artificial	7
2.1.1 O que é Inteligência Artificial?	8
2.2 Computação Evolutiva	9
2.3 Conceito de Algoritmo Genético	11
2.4 Por que utilizar Algoritmos Genéticos?	12
2.5 Definição dos Termos Básicos	14
2.6 Ciclo básico de execução de um AG	15
2.7 Principais Componentes de um AG	15
2.7.1 Codificação ou Representação Cromossômica	16
2.7.2 Populações e Gerações	16

2.7.3	Função de Avaliação	17
2.7.4	Seleção para a reprodução	17
2.7.5	Reprodução	19
2.7.6	Reinserção da população	22
2.7.7	Crítérios de Parada	23
2.8	Problemas dos Algoritmos Genéticos	24
2.9	Algumas Aplicações de Algoritmos Genéticos	26
2.10	Algoritmo Genético Coevolutivo	27
2.11	Ferramentas estatísticas aplicadas na análise de dados obtidos pelos AGs	29
3	Autômatos Celulares	31
3.1	ACs: história, origens e motivação	31
3.2	Conceitos, Definições e Notação	34
3.2.1	Condições de Contorno	36
3.2.2	Modos de Operação	36
3.3	Autômatos Celulares Unidimensionais e Bidimensionais	37
3.3.1	Autômatos Celulares Unidimensionais	37
3.3.2	Autômatos Celulares Bidimensionais	40
3.4	Dinâmica	42
3.5	Computação em ACs	44
3.6	Tarefa de Classificação da Densidade	45
4	Escalonamento de Tarefas em Multiprocessadores	49
4.1	Conceitos e Notações	50
4.2	Algoritmos de escalonamento baseados em heurísticas	57
4.2.1	Métodos de escalonamento de um passo	57
4.2.2	Métodos de escalonamento multipassos	60
4.3	Metaheurísticas aplicadas ao problema do escalonamento de tarefas	63
4.3.1	Algoritmos genéticos aplicados ao escalonamento de tarefas	63

4.3.2	Algoritmo <i>Branch-and-Bound</i> aplicado ao escalonamento de tarefas em multiprocessadores	66
5	Modelos Publicados e Experimentos Iniciais	69
5.1	Escalonamento de Tarefas em Multiprocessadores baseado em Regras de Autômatos Celulares	70
5.1.1	Conceito do escalonador baseado em ACs	70
5.1.2	Modelo de vizinhança linear	72
5.1.3	Modelos de vizinhança não-lineares	73
5.1.4	Arquitetura do escalonador baseado em ACs	79
5.2	Capacidade de generalização das regras de ACs	81
5.3	Grafos de programa utilizados	82
5.4	Sobre os experimentos	86
5.5	Resultados com o modelo de vizinhança linear	88
5.5.1	Experimentos iniciais	88
5.5.2	Teste da capacidade de generalização das regras descobertas através do modelo de vizinhança linear	94
5.6	Resultados com os modelos de vizinhança não-linear e modo de operação sequencial	99
5.6.1	Experimentos reproduzidos e novos	99
5.6.2	Teste da capacidade de generalização das regras descobertas através dos modelos de vizinhança não-lineares	105
6	Novas abordagens propostas para a melhoria da capacidade de generalização de regras	109
6.1	Evolução Conjunta: uma nova abordagem de avaliação para a fase de aprendizagem	110
6.1.1	Vizinhança selecionada e atributo nível dinâmico	112
6.1.2	Vizinhança totalística e atributos co-nível, custo computacional e custo de comunicação	116
6.1.3	Vizinhança Selecionada e atributos co-nível, custo computacional e custo de comunicação	119
6.1.4	Vizinhança Totalística e atributo nível dinâmico	122

6.2	Aprendizagem coevolutiva: uma nova abordagem na busca evolutiva de regras de AC	124
6.2.1	Evolução Conjunta versus Coevolução	129
6.2.2	Coevolução versus Evolução Simples	131
6.3	Resultados com os modelos de vizinhança não-lineares e modo de operação paralelo	133
6.3.1	Evolução Simples	134
6.3.2	Evolução Conjunta	136
6.3.3	Coevolução	140
7	Conclusões e Trabalhos Futuros	147
7.1	Conclusões	147
7.2	Trabalhos Futuros	152
	Referências bibliográficas	154
A	Intervalo de Confiança e Testes de Hipóteses	169
A.1	Intervalo de Confiança	169
A.1.1	Intervalos de confiança de 95% para uma média	169
A.2	Testes de Hipóteses	170
A.2.1	Procedimento geral de testes	172
A.2.2	Teste para uma média	172
A.2.3	Um exemplo de testes de hipóteses	173
B	Principais resultados publicados por Seredynski e colegas	179
B.1	Resultados com o modelo de vizinhança linear	179
B.2	Resultados com os modelos de vizinhança não-linear e o modo de operação seqüencial	182
C	A abordagem Evolução Conjunta e o modelo de vizinhança linear	185
C.1	Experimento	185
D	Alguns resultados detalhados	191
D.1	Modo de operação seqüencial de AC	191
D.1.1	Vizinhança Seleccionada e três atributos	191

D.1.2	Vizinhança Seleccionada e o atributo nível dinâmico	194
D.1.3	Vizinhança Totalística e três atributos	195
D.1.4	Vizinhança Totalística e o atributo nível dinâmico	197
D.2	Modo de operação paralelo de AC	199
D.2.1	Vizinhança Seleccionada e três atributos	199
D.2.2	Vizinhança Seleccionada e o atributo nível dinâmico	201
D.2.3	Vizinhança Totalística e três atributos	202
D.2.4	Vizinhança Totalística e o atributo nível dinâmico	204

Lista de Figuras

2.1	Algumas definições de inteligência artificial, organizadas em quatro categorias [86]. .	8
2.2	Esquema da Taxonomia nos Sistemas baseados em Computação Natural, baseado em [128].	11
2.3	Representação gráfica dos termos básicos para Algoritmos Genéticos: o valor 0,15 corresponde à decodificação de 11101.	15
2.4	Fluxo básico de um algoritmo genético simples	15
2.5	Exemplo de método da roleta para selecionar indivíduos de uma população	19
2.6	Exemplo de <i>crossover</i> ponto-simples.	20
2.7	Exemplo de <i>crossover</i> dois-pontos.	21
2.8	Exemplo de <i>crossover</i> uniforme.	21
2.9	Exemplo de mutação pelo complemento do alelo	22
2.10	Exemplo de mutação por permutação	22
2.11	(a) Estruturas das populações nos AGs e (b) nos AGCs.	28
3.1	Exemplo de um AC unidimensional binário com condição de contorno periódica: (a) reticulado, (b) regra e (c) evolução temporal.	35
3.2	Exemplo de um AC unidimensional binário com condição de contorno não-periódica: (a) reticulado, (b) regra e (c) evolução temporal.	36
3.3	Exemplos de AC unidimensional binário: (a) raio $r = 1$, (b) raio $r = 2$	38
3.4	Diagrama de padrões espaço-temporais de um AC [117].	39
3.5	(a) Exemplo de reticulado de um AC binário bidimensional, (b) a vizinhança de von Neumann e (c) a vizinhança de Moore.	40

3.6	Estrutura periódica no <i>Life</i> semelhante a um relógio [28]	41
3.7	Classificação do comportamento dinâmico dos ACs elementares: (a) Regra 36, (b) Regra 37, (c) Regra 30 e (d) Regra 110 [28].	43
3.8	Diagrama de padrões espaço-temporais para um AC que executou corretamente a TCD [28].	46
4.1	Grafo de sistema representando uma arquitetura com dois processadores	51
4.2	Grafo de programa Gauss18	53
4.3	Um exemplo de solução ótima para o Gauss18 ($T = 44,0$) [96].	53
4.4	Exemplo de <i>clustering</i> : (a) um grafo de programa, (b) um <i>clustering</i> linear e (c) um <i>clustering</i> não-linear [41].	54
4.5	Escalonamento multiprocessador: (a) Grafo de sistema, (b) grafo de programa, (c) e (d) escalonamento utilizando diferentes políticas de escalonamento, para a seguinte alocação nos processadores: tarefas 0, 1 e 2 em P0 e a tarefa 3 em P1.	56
4.6	Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP [41].	59
4.7	Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP [41].	59
4.8	Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP e (d) o escalonamento ótimo, se $c > w$ [41]. . .	61
4.9	Exemplo de um algoritmo DSC [41].	62
4.10	Exemplo de crossover cíclico [89].	65
4.11	Exemplo de permutação simples [89].	65
4.12	Soluções ótimas obtidas com o AG no escalonamento do Gauss18 [89].	66
4.13	Exemplo de árvore de busca: (a) grafo de precedência de tarefas e (b) árvore de busca. . .	67
4.14	Algoritmo Branch-and-Bound.	68
5.1	O conceito do escalonador baseado em ACs proposto em [94].	71

5.2	(a) Exemplo de um reticulado de AC representando a alocação do grafo de programa Gauss18 em um sistema com dois processadores. A vizinhança é linear, de raio $r = 1$. (b) Uma regra de transição para um AC unidimensional binário de raio 1.	73
5.3	Exemplo de reticulado de 18 células associado ao Gauss18, destacando-se as tarefas que fazem parte da vizinhança da tarefa 8.	74
5.4	Vizinhança selecionada para a tarefa 8 do Gauss18.	75
5.5	Vizinhança totalística para a tarefa 8 do Gauss18.	78
5.6	Uma arquitetura de escalonador baseado em AC [96].	80
5.7	Grafos de programa: (a) <i>g18</i> : custo de comunicação igual a 1, (b) <i>tree15</i> : custos de comunicação e computacional igual a 1, (c) <i>g40</i> : custo de comunicação igual a 1 e custo computacional igual a 4 e (d) <i>intree15</i> : custos de comunicação e computacional igual a 1.	83
5.8	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio 1.	89
5.9	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio $r = 2$	89
5.10	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio 3 e modo de operação seqüencial.	90
5.11	Fase de operação do algoritmo de escalonamento para o Gauss18: modo de operação seqüencial e raio $r = 3$: média de T obtida em 1.000 CIs diferentes.	91
5.12	Tempo de execução associado à evolução temporal do AC para o Gauss18.	91
5.13	Fase de aprendizagem do algoritmo de escalonamento para o <i>g18</i>	92
5.14	Fase de operação do algoritmo de escalonamento para o <i>g18</i>	93
5.15	Aplicação das regras descobertas para o <i>g18</i> na fase de operação do algoritmo de escalonamento para o <i>intree15</i>	93
5.16	Fase de aprendizagem do algoritmo de escalonamento para o <i>g40</i> : raio $r = 2$ e os três modos de operação de AC.	95
5.17	Fase de aprendizagem e fase de operação do algoritmo de escalonamento para o Gauss18: Vizinhança Selecionada, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.	100
5.18	Evolução temporal do AC com uma regra que escalona o Gauss18.	101

5.19	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.	102
5.20	Fase de aprendizagem e fase de operação do algoritmo de escalonamento para o Gauss18: Vizinhança Seleccionada, modo de operação seqüencial e atributo nível dinâmico para a formação da vizinhança.	103
5.21	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhança Totalística, modo de operação seqüencial e atributo nível dinâmico para a formação da vizinhança.	103
5.22	Fase de aprendizagem e de operação do algoritmo de escalonamento para o g40: Vizinhança Seleccionada, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.	104
5.23	Fases do algoritmo de escalonamento para o g18: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.	105
5.24	Fases do algoritmo de escalonamento para o tree15: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.	106
6.1	Fase de aprendizagem do algoritmo de escalonamento para o Gauss18 e algumas de suas variações. Utilização da Vizinhança Seleccionada e do atributo nível dinâmico. .	115
6.2	Fase de aprendizagem do algoritmo de escalonamento: Evolução Conjunta considerando a Vizinhança Totalística e os atributos co-nível estático, custo computacional e custo de comunicação.	118
6.3	Fase de aprendizagem do algoritmo de escalonamento utilizando-se a abordagem Evolução Conjunta: Gauss18 + 5 variações, Vizinhança Seleccionada e três atributos diferentes para a formação da vizinhança.	121
6.4	Fase de aprendizagem do algoritmo de escalonamento utilizando-se a abordagem Evolução Conjunta: Gauss18 + 5 variações, Vizinhança Totalística e atributo nível dinâmico para a formação da vizinhança.	123
6.5	Fase de aprendizagem do algoritmo de escalonamento baseado em ACs: processo coevolutivo das populações de regras e grafos de programa.	130

6.6	(a) Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhança Seleccionada, modo paralelo de AC e três atributos diferentes para a formação da vizinhança. (b) Fase de operação.	135
6.7	Fase de aprendizagem do algoritmo de escalonamento: modo de operação paralelo, Evolução Conjunta, Vizinhança Seleccionada com três atributos diferentes.	138
6.8	Fase de aprendizagem do algoritmo de escalonamento baseado em ACs com modo de operação paralelo: processo coevolutivo das populações de regras e grafos de programa.	143
A.1	Duas distribuições amostrais de pesos de embalagens de 1 litro de suco de laranja. As duas são obtidas com amostras de 20 embalagens retiradas das suas respectivas populações. Uma tem média igual a 1080g e a outra não tem. Note que tanto para um caso como para o outro podem haver amostras com peso médio $\bar{x} = 1057g$	175
A.2	Distribuição amostral dos pesos médios de amostras de 20 embalagens.	175
A.3	Distribuição amostral com o valor de $\bar{x} = 1057g$ para a sua amostra aleatória de 20 embalagens.	176
A.4	Distribuição amostral com o valor de $\bar{x} = 1078g$ para a sua amostra aleatória de 20 embalagens.	177
B.1	(a) Exemplo de escalonamento ótimo para o <i>g18</i> . (b) Exemplo de escalonamento ótimo para o <i>intree15</i> [103].	181
C.1	Fase de aprendizagem do algoritmo de escalonamento: evolução conjunta do Gauss18 e 5 variações deste grafo, considerando o modelo de vizinhança linear com o raio $r = 3$.	188

Lista de Tabelas

2.1	Determinação de parâmetros de AGs, segundo [7, 18, 107]	26
3.1	Exemplos de regras totalísticas para um AC unidimensional binário de raio 2.	39
5.1	Grafos de programa e seus tempos de execução T mínimos em um sistema com dois processadores.	84
5.2	Variações do Gauss18	85
5.3	Experimento de referência para determinação dos valores de T mínimos para as quinze variações do Gauss18.	86
5.4	Teste da capacidade generalização das regras descobertas para os grafos de programa Gauss18, <i>g18</i> , <i>g40</i> e <i>tree15</i>	96
5.5	Valores obtidos através da aplicação, na fase de operação, das regras descobertas para o Gauss18 nas quinze variações do mesmo.	98
5.6	Teste da capacidade de generalização das regras de transição.	107
6.1	Evolução Simples e Conjunta, utilizando-se o modo de operação seqüencial, a Vizinhança selecionada e atributo nível dinâmico.	113
6.2	Evolução Simples e Conjunta, considerando-se o modo de operação seqüencial , a Vizinhança Totalística e os atributos co-nível, custo computacional e custo de comunicação	117
6.3	Evolução Simples e Conjunta, considerando-se o modo de operação seqüencial, a Vizinhança Selecionada e os atributos co-nível, custo computacional e custo de comunicação.	120

6.4	Evolução Simples e Conjunta do Gauss18, considerando-se o modo de operação sequencial, a Vizinhança Totalística e o atributo nível dinâmico.	122
6.5	Aplicação das regras obtidas com a Evolução Conjunta, Vizinhança Seleccionada e três atributos diferentes, no escalonamento das 50 variações aleatórias do Gauss18. .	127
6.6	Valores obtidos para 50 variações do Gauss18: abordagem do algoritmo genético coevolutivo.	128
6.7	Aplicação das regras descobertas com a Evolução Conjunta e AGC no escalonamento das 50 variações aleatórias do Gauss18.	130
6.8	Aplicação das regras descobertas com a Evolução Simples e AGC no escalonamento das 50 variações aleatórias do Gauss18.	132
6.9	Tempos, em horas, para execução do algoritmo de escalonamento baseado em ACs para as abordagens Simples, Conjunta e Coevolutiva.	133
6.10	Resultados obtidos com a Evolução Simples do Gauss18 e com a Evolução Conjunta do Gauss18 e 5 variações deste grafo de programa, utilizando-se o modo de operação paralelo, Vizinhança Seleccionada e atributos co-nível estático, custo computacional e custo de comunicação.	137
6.11	Resultados resumidos da comparação Evolução Simples versus Evolução Conjunta no modo de operação paralelo de AC.	139
6.12	Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a Evolução Simples do Gauss18, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	141
6.13	Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	142
6.14	Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a abordagem coevolutiva do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	144

6.15	Aplicação das regras descobertas com o modo de operação paralelo de AC, a Evolução Simples, Conjunta e AGC no escalonamento das 50 variações aleatórias do Gauss18.	145
A.1	Tabela de distribuição de t	171
A.2	Exemplo do cálculo do teste de hipótese: dados amostrais de 20 embalagens de 1 litro de suco	173
C.1	Fase de operação do algoritmo de escalonamento: Evolução Conjunta do Gauss18 mais 5 variações e modelo de vizinhança linear com raio $r = 3$	189
D.1	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.	192
D.2	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.	192
D.3	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Co-evolução do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.	193
D.4	Valores obtidos nas 30 variações do Gauss18: aplicação das regras descobertas com a Co-evolução do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.	193
D.5	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhança Seleccionada com o atributo nível dinâmico e o modo de operação sequencial de AC.	194
D.6	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Seleccionada com o atributo nível dinâmico e o modo de operação sequencial de AC.	194

D.7	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Seleccionada com o atributo nível dinâmico e o modo de operação sequencial de AC.	195
D.8	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com três atributos diferentes e o modo de operação sequencial de AC.	195
D.9	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com três atributos diferentes e o modo de operação sequencial de AC.	196
D.10	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Totalística com três atributos diferentes e o modo de operação sequencial de AC.	196
D.11	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.	197
D.12	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.	197
D.13	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.	198
D.14	Valores obtidos nas 30 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.	198
D.15	Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	199

D.16 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	200
D.17 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Co-evolução do Gauss18 + 9 variações aleatórias, Vizinhaça Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.	200
D.18 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.	201
D.19 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.	201
D.20 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Co-evolução do Gauss18 + 9 variações aleatórias, Vizinhaça Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.	202
D.21 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com três atributos diferentes e o modo de operação paralelo de AC.	202
D.22 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com três atributos diferentes e o modo de operação paralelo de AC.	203
D.23 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Totalística com três atributos diferentes e o modo de operação paralelo de AC.	203
D.24 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.	204

D.25 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.	204
D.26 Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.	205

Glossário

AC - Autômato Celular

AE - Algoritmo Evolutivo

AG - Algoritmo Genético

AGC - Algoritmo Genético Coevolutivo

B&B - Branch-and-Bound

CE - Computação Evolutiva

CI - Configuração Inicial

CP - Critical Path

DS - Dominant Sequence

DSC - Dominant Sequence Clustering

EE - Estratégias Evolutivas

IA - Inteligência Artificial

IC - Inteligência Computacional

LC - Linear Clustering

P0 - Processador 0 (zero)

P1 - Processador 1

PE - Programação Evolutiva

PG - Programação Genética

SA - Simulated Annealing

SC - Sistemas Classificadores

SIA - Sistemas Imunológicos Artificiais

TCD - Tarefa de Classificação da Densidade

TS - Tabu Search

VA - Vida Artificial

Capítulo 1

Introdução

A principal solução proposta para a melhoria da performance dos computadores nos últimos anos tem sido o paralelismo. Entretanto, a programação e o gerenciamento de máquinas paralelas ainda são tarefas muito difíceis [98]. O problema do escalonamento em multiprocessadores é conhecido por ser NP-completo em sua forma geral [13, 36, 96] e, por isso, ele consiste em um desafio para muitos pesquisadores. A maioria dos algoritmos de escalonamento retornam soluções sub-ótimas, e alguns deles estão aptos a encontrar soluções ótimas apenas para casos especiais do problema.

Enquanto grande parte dos algoritmos de escalonamento conhecidos são seqüenciais, heurísticas baseadas em diferentes plataformas matemáticas e metaheurísticas baseadas em mecanismos observados na natureza têm sido introduzidos [104]. Metaheurísticas ou técnicas inteligentes tais como, o *simulated annealing* (SA), os *algoritmos genéticos* (AGs) e as *redes neurais* têm sido aplicadas com sucesso nessa tarefa [53, 87]. A utilização de técnicas de busca estocástica, derivadas da natureza, abrem novas possibilidades para se alcançar soluções de boa qualidade para determinados problemas onde o alcance destas soluções, através de métodos seqüenciais e determinísticos, é inviável.

Resultados recentes [96, 104] mostraram que, combinados, os autômatos celulares (ACs) [120] e os algoritmos genéticos (AGs) [43] podem ser efetivamente usados para projetar algoritmos paralelos e distribuídos para resolver problemas complexos de classificação [71], sincronização [34] e alguns problemas do tipo NP-completo [39]. Os AGs também podem ser usados no projeto de algoritmos de escalonamento em sistemas multiprocessadores [95, 96].

Os autômatos celulares (ACs) são sistemas dinâmicos nos quais tempo e espaço são discretos.

Estes sistemas são formados por componentes simples e idênticos (conhecidos como células) com conectividade local, e por uma regra de transição de estados (ou função de transição) que determina qual o próximo estado do AC a partir de seu estado atual. Uma de suas características interessantes é que, embora haja uma interação local entre as células, um comportamento complexo global pode emergir. Por isso, os ACs são freqüentemente usados para modelar fenômenos do mundo real [91] e também são considerados como um modelo de computação altamente paralela e distribuída [68].

Dentre as habilidades dos ACs que mais atraem o interesse dos pesquisadores, está a sua capacidade em realizar computações baseadas em interações puramente locais, onde as células são de uma lógica simples. No entanto, a compreensão de como estes sistemas são capazes de computar é algo ainda extremamente vago e os pesquisadores têm se empenhado em buscar métodos para viabilizar a sua programação.

O principal problema relacionado ao uso dos ACs é que o espaço formado pelas regras de transição, que representam as possíveis soluções de um problema, é tipicamente de alta cardinalidade. Entretanto, trabalhos recentes abriram novas possibilidades ao apresentar a aplicação da computação evolutiva para a busca de regras de ACs [23, 101]. Uma das técnicas evolutivas utilizadas nessa busca são os algoritmos genéticos (AGs) [43]. Os AGs são probabilísticos e fornecem um mecanismo de busca paralela e adaptativa baseado no princípio da sobrevivência dos indivíduos mais aptos e na reprodução, inspirados no princípio Darwiniano de seleção natural e reprodução genética [43].

O algoritmo de escalonamento proposto em [94] e utilizado como base neste trabalho apresenta duas fases. Na fase de aprendizagem de regras, a busca de regras de ACs capazes de escalonar um programa paralelo é conduzida com a utilização de um algoritmo genético. Na fase de operação, espera-se que para qualquer alocação inicial das tarefas entre dois processadores, a regra do AC seja capaz de encontrar alocações das tarefas entre os processadores provendo o valor mínimo (ou próximo) do tempo total de execução T .

Ao utilizar autômatos celulares para o problema do escalonamento, é preciso projetar uma estrutura de AC e de vizinhança das células com base no programa paralelo a ser escalonado. Neste trabalho, foram utilizados três tipos de vizinhança para o AC: uma vizinhança linear [102] e outras duas não lineares [96], denominadas *Vizinhança Seleccionada* e *Vizinhança Totalística*.

1.1 Objetivo

O objetivo do nosso trabalho é utilizar o algoritmo de escalonamento baseado em autômatos celulares e algoritmos genéticos proposto em [94] na busca de regras de transição de ACs que possuam uma capacidade de generalização desejável. Essa capacidade refere-se ao fato de que essas regras ao serem aplicadas em diferentes programas paralelos, são capazes de encontrar alocações destes programas que provêm o valor ótimo ou sub-ótimo do tempo total de execução.

1.2 Justificativa

A maioria dos algoritmos de escalonamento não extraem, conservam e reutilizam qualquer conhecimento sobre o problema ao resolverem instâncias do problema de escalonamento. Para exemplificar esta situação, suponhamos a aplicação de um escalonador baseado apenas em um AG. Para a execução do escalonador, uma população inicial aleatória de possíveis soluções é criada. A população é evoluída com a utilização de operadores genéticos até que uma solução seja encontrada, ou um número máximo de gerações tenha sido atingido. Para encontrar uma solução de um novo programa paralelo, ou um programa que é uma variação (modificação) do programa paralelo anterior, ou uma composição de programas paralelos resolvidos anteriormente, é necessário iniciar o processo de busca novamente, isto é, executar o AG novamente na busca de soluções para este novo programa paralelo.

A motivação desse trabalho é elaborar novas abordagens para a fase de aprendizagem no algoritmo de escalonamento baseado em autômatos celulares e algoritmos genéticos. Espera-se que com essas abordagens, as regras de transição evoluídas pelos AGs possam utilizar algum conhecimento sobre o programa paralelo para o qual elas foram evoluídas, para escalonar variações (modificações) deste programa ou até mesmo outros programas paralelos. Sendo assim, essas regras podem ser consideradas como regras com capacidade de generalização. Dessa forma, não é necessário realizar um novo processo de busca evolutiva para cada novo programa paralelo que se deseja escalonar. As regras evoluídas pelo AG podem ser aplicadas diretamente no escalonamento de programas paralelos para encontrar alocações ótimas (ou sub-ótimas) das tarefas nos processadores.

É interessante ressaltarmos que essa capacidade de generalização das regras evoluídas na fase de aprendizagem é crucial para o escalonamento baseado em ACs. Todo o esforço computacional

empregado na descoberta das regras de ACs só se justifica se elas puderem ser reutilizadas em novos problemas. Do contrário, um AG pode ser utilizado diretamente na busca da configuração ótima de cada programa independentemente, sem a necessidade de envolvermos os ACs no modelo. A idéia de se utilizar as regras de ACs é justamente pela possibilidade de reaproveitamento das mesmas em novos problemas, sem a necessidade de um novo processo de aprendizagem evolutivo, que exige um grande esforço computacional.

1.3 Contribuições

Duas novas abordagens são apresentadas nesse trabalho para o algoritmo de escalonamento baseado em ACs e AGs, mais especificamente para a fase de aprendizagem de regras de transição de ACs deste algoritmo.

A abordagem Evolução Conjunta [113] utiliza mais de um programa paralelo na fase de aprendizagem para evoluir as regras de ACs através de um AG padrão. Na fase de operação, as regras evoluídas são aplicadas aos programas paralelos que participaram na fase anterior e em outros programas que não participaram desta fase. Os resultados encontrados [112, 113] com esta abordagem mostraram que as regras evoluídas possuem uma maior capacidade de generalização quando comparadas às regras evoluídas utilizando apenas um programa paralelo na fase de aprendizagem (Evolução Simples).

A segunda abordagem investigada foi a utilização de um ambiente coevolutivo [48] aplicado também na fase de aprendizagem de regras. Nesta abordagem, detalhada no Capítulo 6, os resultados mostraram que as regras evoluídas apresentam uma maior capacidade de generalização quando comparadas tanto às regras evoluídas utilizando a Evolução Simples quanto às regras encontradas através da Evolução Conjunta.

Outra investigação inédita conduzida nesse trabalho é a utilização apenas do atributo nível dinâmico na formação da vizinhança do AC. Os experimentos com esse atributo foram realizados tanto na abordagem original proposta por Seredynski e colegas, que chamamos de Evolução Simples, quanto nas novas abordagens Evolução Conjunta e Co-evolução.

1.4 Estrutura do Trabalho

Para uma melhor compreensão do leitor, esta dissertação está organizada como segue.

O Capítulo 2 refere-se ao algoritmo genético (AG), apresentando os principais conceitos, as características fundamentais, os elementos e a estrutura básica desse método. São apresentados os principais operadores genéticos (seleção, *crossover* e mutação) e os principais métodos utilizados para cada um deles. Além do AG mais simples, também conhecido como AG padrão, apresentamos um resumo dos algoritmos genéticos coevolutivos, enfatizando as principais diferenças em relação ao AG padrão. Uma breve discussão sobre a aplicação do Teste de Hipótese [73] na avaliação dos experimentos evolutivos também é apresentada.

O Capítulo 3 refere-se aos autômatos celulares (ACs). É apresentado um resumo sobre a história dos ACs e as contribuições de alguns de seus principais pesquisadores. Os conceitos básicos e elementos que compõem a modelagem de um AC também são fornecidos. A resolução da tarefa computacional conhecida por Tarefa da Classificação da Densidade [68, 71, 72] também é apresentada, onde são aplicadas técnicas evolutivas para a busca de regras de ACs aptas a resolver esse problema.

O Capítulo 4 apresenta os conceitos fundamentais para o escalonamento de tarefas em sistemas multiprocessadores. Alguns algoritmos tradicionais utilizados para escalonar programas paralelos são apresentados. Apresentamos também um exemplo de AG simples utilizado para escalonar um programa paralelo.

O Capítulo 5 apresenta os conceitos, a arquitetura e os modelos de vizinhança, propostos por Seredynski e colegas, utilizados no escalonador baseado em ACs. Também é discutida a capacidade de generalização de regras de ACs para o escalonamento e porque ela é fundamental nos algoritmos de escalonamento baseados em ACs. São apresentados os principais grafos de programa utilizados nesse trabalho, e resultados experimentais obtidos com o modelo de vizinhança linear e com os modelos não-lineares, utilizando-se o modo de operação seqüencial de AC. Estes resultados representam experimentos preliminares que realizamos.

O Capítulo 6 apresenta as novas abordagens propostas para a melhoria da capacidade de generalização das regras e os resultados dos experimentos realizados com estas abordagens e o modo de operação seqüencial de AC, os quais serviram de base para o direcionamento e as conclusões do trabalho. Também são apresentados alguns experimentos que realizamos com o modo de operação paralelo

de AC, tanto com a abordagem Evolução Simples quanto com as novas abordagens propostas.

O Capítulo 7 apresenta as conclusões deste trabalho, confrontando os resultados experimentais obtidos em relação ao objetivo do trabalhos e alguns trabalhos futuros são propostos.

Capítulo 2

Algoritmos Genéticos

Neste capítulo, discutiremos os Algoritmos Genéticos (AGs), apresentando os principais conceitos, as características fundamentais, a estrutura básica e os principais elementos utilizados na modelagem de um algoritmo genético padrão. Algumas aplicações de algoritmos genéticos são apresentadas. Também é discutido o Algoritmo Genético Coevolutivo (AGC), com ênfase nas diferenças apresentadas em relação ao algoritmo genético padrão. Ao final do capítulo, apresentamos uma introdução ao Intervalo de Confiança e ao Teste de Hipótese, medidas estatísticas comumente utilizadas na avaliação comparativa de experimentos evolutivos.

2.1 Inteligência Artificial

A inteligência pode ser definida como a capacidade mental de raciocinar, planejar, resolver problemas, abstrair idéias, compreender linguagens e aprender. Tais capacidades mentais são muito importantes para nós, seres humanos. Entender como pensamos não é uma tarefa trivial. O campo da Inteligência Artificial (IA), além de tentar entender como pensamos, tenta construir entidades capazes de demonstrar inteligência. Estas entidades podem ser, por exemplo, computadores. Capacitar um computador de um comportamento inteligente significa fazer com que ele realize atividades que somente um ser humano seria capaz de efetuar [45].

A IA é uma ciência recente. As primeiras pesquisas começaram logo após a Segunda Guerra Mundial, e o próprio nome foi cunhado em 1956 [86].

2.1.1 O que é Inteligência Artificial?

A inteligência artificial surgiu na década de 50, e foi Alan Turing quem primeiro articulou uma visão completa da IA em seu artigo intitulado “Computing Machinery and Intelligence” [86, 109]. Nesse artigo, ele apresentou o Teste de Turing, aprendizagem de máquina, algoritmos genéticos e aprendizagem por reforço.

O objetivo da IA é desenvolver sistemas para realizar tarefas que, devido a sua natureza cognitiva, são melhor realizadas por seres humanos que por máquinas, ou não possuem uma solução algorítmica viável pela computação convencional [86]. A Figura 2.1 apresenta algumas definições de inteligência artificial, organizadas em quatro categorias diferentes. Essas definições variam ao longo de duas dimensões principais. As que estão na parte superior da tabela se relacionam a *processos de pensamento e raciocínio*, enquanto as definições da parte inferior se referem ao *comportamento*. As definições do lado esquerdo medem o sucesso em termos de fidelidade ao desempenho humano, enquanto as definições do lado direito medem o sucesso comparando-o a um conceito *ideal* de inteligência, chamado **racionalidade**. Um sistema é racional se “faz tudo certo”, com os dados que tem [86]. Historicamente, todas as quatro categorias (abordagens) para o estudo de inteligência artificial têm sido seguidas e, cada grupo, tem ao mesmo tempo desacreditado e ajudado o outro [86].

Sistemas que pensam como seres humanos	Sistemas que pensam racionalmente
“O novo e interessante esforço para fazer os computadores pensarem... máquinas com <i>mentes</i> , no sentido total e literal.” (Haugeland, 1985)	“O estudo das faculdades mentais pelo uso de modelos computacionais.” (Charniak e McDermott, 1985)
“[Automatização de] atividades que associamos ao pensamento humano, atividades como a tomada de decisões, a resolução de problemas, o aprendizado...” (Bellman, 1978)	“O estudo das computações que tornam possível perceber, raciocinar e agir.” (Winston, 1992)
Sistemas que atuam como seres humanos	Sistemas que atuam racionalmente
“A arte de criar máquinas que executam funções que exigem inteligência quando executadas por pessoas.” (Kurzweil, 1990)	“A Inteligência Computacional é o estudo do projeto de agentes inteligentes.” (Poole <i>et al.</i> , 1998)

Fig. 2.1: Algumas definições de inteligência artificial, organizadas em quatro categorias [86].

Dentro da IA existem várias técnicas diferentes, cada qual com a sua aplicabilidade e utilidade, dentre as quais se destaca a *Inteligência Computacional* (IC). Devido à dificuldade existente para que

os sistemas inteligentes tradicionais resolvam determinadas classes de problemas, como reconhecimento de padrões ou sistemas reativos, intensificaram-se as pesquisas no campo da IC [128]. Dentre as técnicas da IC [76] destacam-se as *Redes Neurais Artificiais*, a *Lógica Fuzzy*, os *Agentes Autônomos* e a *Computação Evolutiva* (CE). Nessa dissertação, a técnica da IC utilizada é a computação evolutiva (mais especificamente os algoritmos genéticos), descrita na próxima seção.

2.2 Computação Evolutiva

A Computação Evolutiva (CE) surgiu no final dos anos 60, quando John Holland começou a estudar a possibilidade de incorporar os mecanismos naturais, baseados nos princípios Darwinianos da seleção natural e sobrevivência, para a resolução de problemas de inteligência artificial, os quais já tinham encontrado solução na natureza mas não apresentavam uma abordagem satisfatória em sistemas computacionais. Como resultado de sua pesquisa, Holland lançou seu livro *Adaptation in Natural and Artificial Systems* [49] o qual é considerado o ponto de partida da CE. Uma revisão sobre a história da Computação Evolutiva é apresentada em [6].

Muitos dos problemas computacionais requerem a busca de soluções em um imenso espaço de possibilidades, o que exige um alto esforço computacional. Por exemplo, a busca por um conjunto de equações que identificarão os altos e baixos do mercado financeiro, ou um conjunto de regras que controlarão um robô em seu ambiente de navegação. Outros problemas computacionais frequentemente requerem que um sistema seja adaptativo, isto é, que ele continue com seu comportamento adequado mesmo diante de mudanças no ambiente. Programar soluções computacionais para estes tipos de problemas não é algo trivial, visto a complexidade dos mesmos. No passado, praticantes e pesquisadores da inteligência artificial acreditavam que seria direta a codificação de regras que atribuiriam inteligência a um programa. Atualmente, muitos pesquisadores acreditam que tais regras são muito complexas para serem codificadas manualmente, em uma topologia do tipo *top-down*¹. Ao invés disso, eles acreditam que o melhor caminho para a inteligência artificial e outros problemas computacionais de difícil solução é uma abordagem *bottom-up*, na qual são escritas apenas as regras mais simples e é provido ao sistema meios para se adaptar. Um comportamento complexo, como a

¹A abordagem top-down dá ênfase ao planejamento e entendimento *completo* do sistema. É um estilo de programação no qual o projeto inicia pela especificação das partes complexas e então as divide em partes sucessivamente menores.

inteligência, emergirá da aplicação paralela e da interação destas regras. As redes neurais artificiais [51] são um exemplo desta filosofia; a computação evolutiva é outro.

A computação evolutiva baseia-se fundamentalmente no uso de Algoritmos Evolutivos (AEs), cujo propósito é conduzir uma busca estocástica, fazendo evoluir um conjunto de indivíduos (representações de soluções do problema) e selecionando aqueles mais adequados. Os AEs permitem aos indivíduos sobreviver e reproduzir em seus ambientes, podendo assim serem considerados métodos para adaptação dos indivíduos a mudanças no ambiente. Durante o processo evolutivo, alguns operadores genéticos (mutação e cruzamento) são aplicados aos indivíduos, após o processo de seleção “natural”, onde o indivíduo mais adaptado ao ambiente tem maior chance de reproduzir, garantindo assim a propagação do seu material genético para as futuras gerações.

Um AE é considerado mais eficiente quanto melhor seja seu desempenho na solução de um determinado problema, independentemente de sua fidelidade aos conceitos biológicos. Na verdade, a maioria dos algoritmos que seguem este enfoque são simples do ponto de vista biológico mas, mesmo assim, apresentam-se como poderosas e eficientes ferramentas de busca.

Devido ao alto esforço computacional exigido, somente a partir dos anos 80, com o surgimento de computadores de alto desempenho e baixo custo, foi viável a utilização prática de sistemas com técnicas de computação evolutiva. A partir de então, esta técnica foi bastante utilizada na resolução de certos problemas de engenharia e das ciências sociais que até então não encontravam forma prática de implementação em modelos computacionais. Neste período, surgem alguns trabalhos de Goldberg [43] e de Davis [25] que propõem soluções para problemas da vida real.

O que é crucial para o sucesso do algoritmo evolutivo é como as soluções candidatas são representadas em estruturas de dados. Isto é conhecido como a representação do problema. A solução Darwiniana de problemas de otimização é possível se e somente se o problema é “codificado” de maneira a realizar o processo de seleção-cruzamento-mutação de forma eficiente. A representação do problema é a maneira de codificar o mesmo de forma que variações aleatórias e a seleção podem levar para à solução. Esta representação deve garantir que a seleção, mutação e/ou reprodução possam produzir a adaptação.

A computação evolutiva é uma das grandes áreas da inteligência computacional, assim como as redes neurais artificiais e os sistemas fuzzy, que por sua vez fazem parte da computação natural, que

aglomera as técnicas computacionais que têm na natureza sua principal fonte de inspiração [128]. Em [76], Pal e Wang dividiram a computação evolutiva, como podemos visualizar na Figura 2.2, em cinco paradigmas básicos: Algoritmos Genéticos (AG), Programação Evolutiva (PE), Estratégias Evolutivas (EE), Sistemas Classificadores (SC) e Programação Genética (PG).

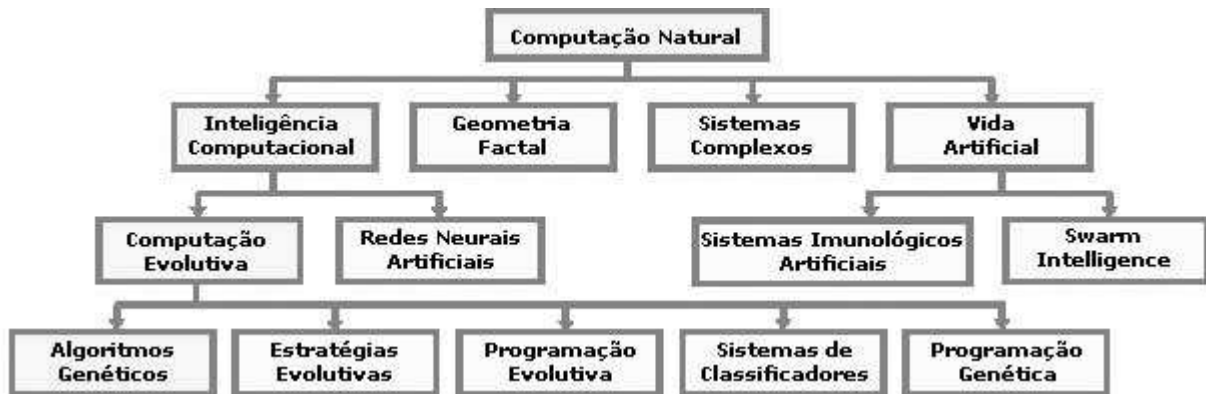


Fig. 2.2: Esquema da Taxonomia nos Sistemas baseados em Computação Natural, baseado em [128].

Dentre as técnicas acima citadas, os algoritmos genéticos são as mais estudadas e difundidas devido à sua flexibilidade, simplicidade de implementação e eficácia na realização de busca global em ambientes diversos [107]. Na próxima seção, descrevemos com mais detalhes os conceitos, os principais elementos, a estrutura básica e algumas aplicações dos AGs, que são os AEs utilizados no algoritmo de escalonamento baseado em autômatos celulares, discutido no Capítulo 5.

2.3 Conceito de Algoritmo Genético

“Algoritmos genéticos são métodos computacionais de busca, baseados nos mecanismos da evolução natural e na genética natural. Eles combinam a sobrevivência do melhor adaptado dentre estruturas formadas por seqüências de *bits*, com um troca de informação randômica e estruturada para formar um algoritmo computacional com algum “*faro*” inovador da busca humana (...) Apesar de aleatórios, os algoritmos genéticos não são uma simples caminhada aleatória. Eles exploram eficientemente informações históricas para especular novos pontos de busca com um aumento esperado de performance” [43].

“Em AGs uma população de possíveis soluções para o problema em questão evolui de acordo

com operadores probabilísticos concebidos a partir de metáforas biológicas, de modo que há uma tendência de que, na média, os indivíduos representem soluções cada vez melhores à medida que o processo evolutivo continua” [107].

AG é uma técnica de busca e otimização global, inspirada nos mecanismos da seleção natural e da reprodução genética, proposta por Darwin. Ela utiliza uma estratégia de busca paralela e estruturada, porém aleatória, que é voltada em direção ao reforço da busca de pontos de "alta aptidão", ou seja, pontos nos quais a função a ser minimizada (ou maximizada) tem valores relativamente baixos (ou altos) [88].

Um algoritmo genético caracteriza-se como um procedimento iterativo no qual cada iteração é chamada de *geração*. Na primeira geração, é gerada uma população formada por um conjunto aleatório de indivíduos que podem ser vistos como possíveis soluções para um determinado problema. Durante o processo evolutivo, esta população é avaliada, sendo associada a cada indivíduo uma nota (*fitness*), calculada através de uma *função de aptidão*, que reflete a sua habilidade de adaptação a um determinado ambiente. Seguindo o Darwinismo, os indivíduos mais adaptados têm maior chance de sobreviver, enquanto os menos adaptados tendem a ser descartados. Os indivíduos escolhidos pelo operador de *seleção* podem sofrer modificações em suas características fundamentais através de outros dois operadores: *mutação* e *crossover* (cruzamento ou recombinação genética). Desta maneira, são gerados descendentes para a próxima geração. Este processo, chamado de *reprodução*, é repetido até que uma solução satisfatória seja encontrada ou um número máximo de gerações seja atingido. Embora muita ênfase seja dada aos três operadores genéticos acima mencionados, a representação do problema e a função de aptidão são os aspectos mais importantes de qualquer AG, porque eles são dependentes do problema [21].

2.4 Por que utilizar Algoritmos Genéticos?

Além da fácil implementação, os AGs são bastante flexíveis e eficazes ao realizar a busca global por soluções em diversos ambientes diferentes. Por esta razão, eles têm sido aplicados na resolução, entre outros, de problemas de otimização combinatorial². A otimização combinatorial envolve um grande

²Uma definição informal para o domínio da otimização combinatorial é problemas onde o conjunto de *soluções práticas* é discreto (ou pode ser reduzido para um discreto), e o objetivo é encontrar a melhor solução possível.

número de possíveis soluções, fazendo com que o uso das técnicas de enumeração (por exemplo, *branch and bound*, *cutting plane*, ou *programação dinâmica*) seja desacreditado [21].

As técnicas de busca e otimização tradicionais iniciam-se com um único candidato a solução do problema que, iterativamente, é manipulado utilizando algumas heurísticas (estáticas) associadas ao problema a ser solucionado. Geralmente, estes processos heurísticos não são algorítmicos e sua simulação em computadores pode ser algo inviável. Apesar destes métodos não serem suficientemente robustos, isto não significa que eles sejam inúteis. Na prática, eles são amplamente utilizados, com sucesso, para a resolução de inúmeros problemas.

Em [43] são apresentados quatro princípios básicos que diferem os AGs da maioria dos procedimentos de busca e otimização:

1. AGs operam num espaço de soluções codificadas, e não no espaço de busca diretamente.
2. AGs operam num conjunto (população) de pontos, e não a partir de um ponto isolado.
3. AGs necessitam somente de informação sobre o valor de uma função objetivo para cada membro da população e não requerem derivadas ou qualquer outro tipo de conhecimento.
4. AGs usam regras de transição probabilísticas, e não regras determinísticas.

O primeiro princípio básico citado não é sempre válido nas aplicações atuais de AGs. Embora na maioria das aplicações iniciais fosse utilizada uma codificação para representar o problema, os AGs também podem ser utilizados diretamente no espaço de busca. Nas primeiras aplicações dos AGs, a codificação binária era normalmente utilizada para representar o problema, diferentemente de outros métodos baseados em busca estocástica, onde os problemas de otimização são resolvidos através da representação original do espaço de busca [21].

Outros procedimentos de otimização que, como os AGs, vão além de uma simples busca local são o *Simulated Annealing* (SA) [1] e o método de meta-heurísticas *Tabu Search* (TS) [42]. O TS compartilha, com os AGs, os dois últimos princípios citados anteriormente [27]. Porém, contrário aos AGs, ele utiliza uma busca determinística, baseada na vizinhança com a anulação de ótimos locais. O SA utiliza um função de probabilidade que permite um “movimento” para uma solução pior com uma probabilidade decrescente, à medida que a busca continua. Quando uma função objetivo

tem muitos mínimos locais igualmente bons, onde quer que o ponto de partida esteja, uma pequena perturbação aleatória pode evitar o pequeno mínimo local e alcançar um dos bons mínimos, sendo este um problema apropriado para a utilização de SA. Entretanto, SA é menos conveniente para um problema em que há um mínimo global que é muito melhor que todos os outros mínimos locais. Neste caso, é muito importante encontrar aquele “vale”. Assim, é melhor gastar menos tempo na melhoria de qualquer conjunto de parâmetros e mais tempo trabalhando de forma a examinar diferentes regiões do espaço. Isto é o que os AGs fazem melhor.

2.5 Definição dos Termos Básicos

Para um melhor entendimento do leitor, definiremos aqui alguns termos básicos [43, 69] utilizados no contexto dos AGs e que serão utilizados nesta dissertação:

- Cromossomo ou Indivíduo: estrutura de dados que representa uma das possíveis soluções do problema;
- População: conjunto de cromossomos ou indivíduos;
- Gene: seqüência de sub-estruturas elementares que formam o cromossomo, ou seja, a característica ou a unidade básica do cromossomo;
- Alelo: instância de um gene, ou valor da característica;
- Locus: posição de um gene dentro do cromossomo;
- Fenótipo: constitui o conjunto de valores dos parâmetros que formam a solução, no domínio da definição do problema. Ou seja, o conjunto de características decodificadas;
- Genótipo: conjunto de valores codificados que representam a solução, ou seja, o conjunto de características. Um genótipo é uma instância do cromossomo.

Para facilitar a melhor compreensão, a Figura 2.3 apresenta uma representação gráfica dos termos acima citados.

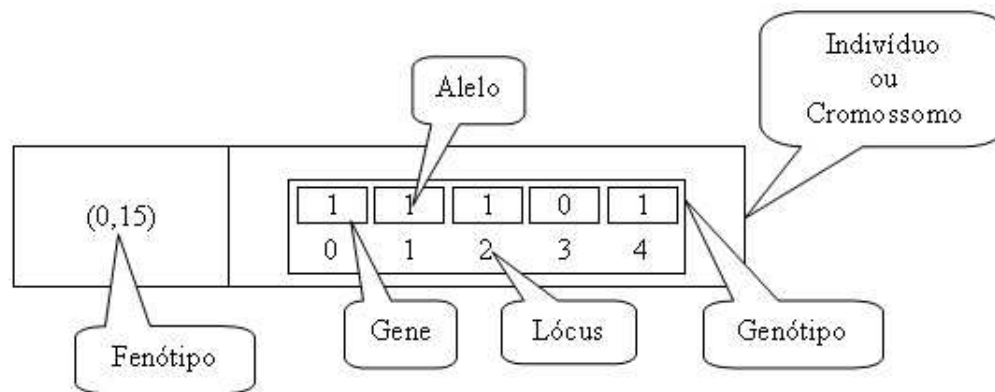


Fig. 2.3: Representação gráfica dos termos básicos para Algoritmos Genéticos: o valor 0,15 corresponde à decodificação de 11101.

2.6 Ciclo básico de execução de um AG

A Figura 2.4 apresenta o ciclo básico de execução de um AG, de acordo com [107].

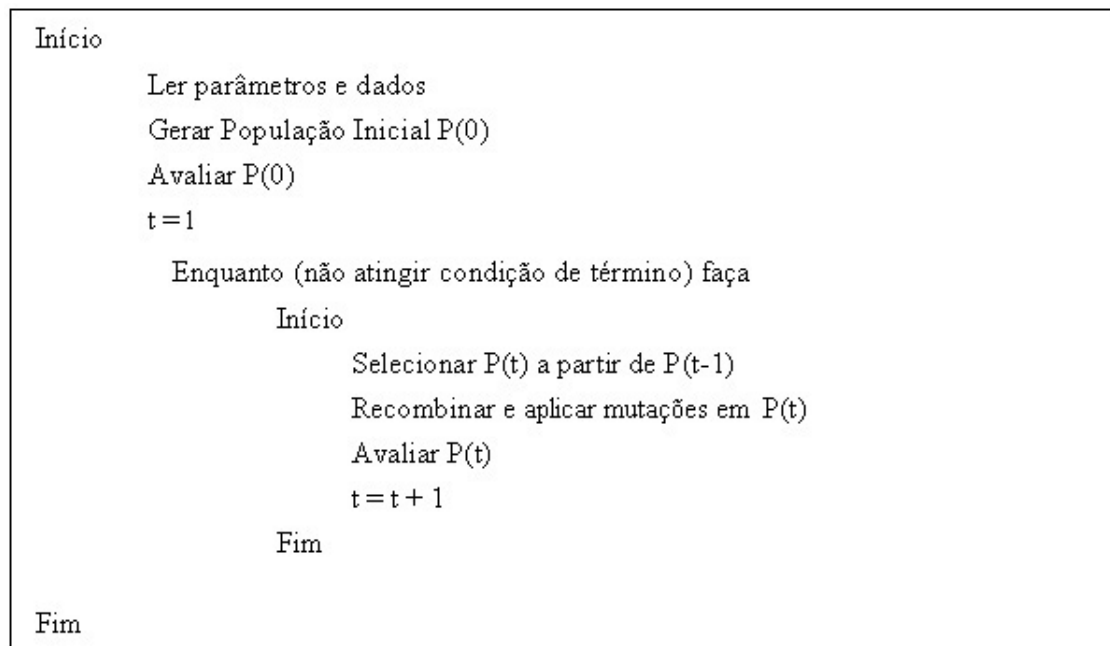


Fig. 2.4: Fluxo básico de um algoritmo genético simples

2.7 Principais Componentes de um AG

A seguir, detalhamos os pontos principais para a modelagem de um algoritmo genético.

2.7.1 Codificação ou Representação Cromossômica

O primeiro e fundamental passo para a aplicação de AGs a um problema específico é decidir que tipo de genótipo o problema necessita. Ou seja, decidir a maneira apropriada de representar uma possível solução x no espaço de busca como uma sequência de símbolos S gerados a partir de um alfabeto finito A . O alfabeto binário $A = \{0,1\}$ é frequentemente usado, embora outras representações possam ser usadas, incluindo codificações baseadas em caracteres e valores reais. Também no caso mais simples, o comprimento do indivíduo (ou cromossomo) é constante durante todo o processo evolutivo. Cada sequência S corresponde a um cromossomo, e cada elemento $s \in S$ é equivalente a um gene. Uma vez que cada gene pode assumir um valor de A , cada elemento de A é um alelo. O índice de cada elemento $s \in S$ corresponde ao seu locus gênico. A Figura 2.3 ilustra uma representação cromossômica formada por 5 genes, que utiliza o alfabeto binário.

2.7.2 Populações e Gerações

Uma vez definida a representação cromossômica, o próximo passo é a geração de um conjunto de possíveis soluções chamadas de soluções candidatas (ou indivíduos). $P(t)$ representa um conjunto de indivíduos em uma dada geração t . Sendo assim, $P(0)$ representa a população inicial de um algoritmo genético, que pode ser gerada aleatoriamente ou através de algum processo heurístico. É de suma importância que $P(0)$ cubra a maior área possível do espaço de busca. Como o processo evolutivo dos AGs simula o processo evolutivo da Natureza, para ocorrer evolução é preciso que haja variedade, pois é necessário que indivíduos tenham diferentes graus de aptidão (ou adaptação) para que possa ocorrer a seleção natural.

Os AGs são algoritmos iterativos e, a cada iteração, a população é modificada, sendo que os indivíduos de melhor aptidão tendem a permanecer e os de pior aptidão tendem a desaparecer, durante o processo evolutivo. Por isso, cada iteração do AG é chamada de geração, embora nem todos os indivíduos de uma nova geração sejam efetivamente “filhos” dos indivíduos da população da iteração anterior [27]. Na maioria dos AGs, o tamanho da população é fixo por questões de simplicidade e facilidade de implementação. A cada geração, os indivíduos da população evoluem através da aplicação dos operadores genéticos.

2.7.3 Função de Avaliação

Cada indivíduo é avaliado e lhe é atribuído um valor de aptidão depois da criação da população inicial. Nas demais gerações, a aptidão é recalculada para todos os indivíduos que formarem a população corrente, a não ser em problemas nos quais o tipo de avaliação utilizada é invariável de uma geração para outra. Nesse caso, a aptidão de indivíduos remanescentes de uma geração anterior não precisa ser recalculada. A avaliação é responsável por “dirigir” o processo de busca.

As funções de avaliação são específicas para cada problema e a especificação de uma apropriada função de avaliação é crucial para a correta operação de um AG [85]. Esta função permite a determinação do valor de adaptabilidade de cada indivíduo na população, isto é, o quão bem uma possível solução satisfaz ao problema original. Quanto melhor o valor da função de avaliação (aptidão) associado a um indivíduo, maiores são as suas chances de sobreviver e reproduzir, passando seu material genético às futuras gerações [27].

2.7.4 Seleção para a reprodução

Realizada a avaliação dos indivíduos, o processo de seleção em AGs seleciona indivíduos para a reprodução. Este processo é inspirado na seleção natural. A seleção é baseada na aptidão dos indivíduos, sendo que os mais aptos têm maior probabilidade de serem escolhidos para a reprodução.

Vários são os esquemas possíveis para se realizar a seleção. No mais utilizado, gera-se uma população temporária de indivíduos extraídos com probabilidade proporcional à aptidão relativa de cada indivíduo na população. Se f_i é a avaliação do indivíduo i na população corrente, a probabilidade p_i do indivíduo i ser selecionado é proporcional a:

$$p_i = \frac{f_i}{\sum_{j=1}^N f_j} \quad (2.1)$$

onde N é o número de indivíduos na população.

A seleção é o operador responsável por determinar as características de convergência dos AGs [15, 44]. A pressão seletiva (ou intensidade de seleção) é o grau a que os melhores indivíduos são favorecidos [8]. A definição de intensidade de seleção empregada em genética é a variação na ap-

tidão média da população induzida pelo método de seleção [14]. Os melhores indivíduos são mais favorecidos à medida que a pressão seletiva aumenta.

A taxa de convergência de um AG é amplamente determinada pela magnitude da pressão seletiva [21]. Ou seja, uma maior pressão seletiva implica em maiores taxas de convergência. Se a pressão seletiva for muito baixa, a taxa de convergência será lenta, e o AG, desnecessariamente, tomará um caminho mais longo para encontrar uma solução de alta qualidade. Se a pressão seletiva for muita alta, é muito provável que o AG, prematuramente, convergirá para uma solução ruim. De fato, mecanismos de seleção devem preservar a diversidade da população, além de prover uma pressão seletiva adequada. Alguns dos mecanismos de seleção mais utilizados são o método da roleta, a seleção por torneio e a seleção por truncamento.

O Método da Roleta é o processo de seleção mais conhecido e utilizado. Na verdade, ele corresponde à seleção com probabilidade proporcional à aptidão, dada pela Equação 2.1. Entretanto, utiliza-se uma roleta como metáfora do sorteio probabilístico. Assim, os indivíduos de uma população são escolhidos para participar da reprodução através de um sorteio de roleta, sendo que cada indivíduo é representado nesta roleta por uma ou mais casas (cavidades). Para cada seleção de um indivíduo que irá reproduzir, a roleta é “girada” e o indivíduo relacionado à casa sorteada é selecionado. Uma característica fundamental da roleta é que o número de casas correspondentes a um indivíduo é proporcional à sua aptidão, de modo que indivíduos mais adaptados têm maior probabilidade de serem selecionados [5]. A Figura 2.5 ilustra este método, onde são apresentados oito indivíduos que compõem a população em uma determinada geração, e seus respectivos valores de aptidão (*fitness*). A roleta possui 100 casas, valor correspondente à somatória dos valores de aptidão de toda a população. Como o indivíduo C1 tem valor de aptidão igual a 12, ele ocupa 12 casas na roleta. O indivíduo C2 tem valor de aptidão igual 13 e por isso ocupa outras 13 casas na roleta. A distribuição de casas na roleta é feita de forma similar para os outros seis indivíduos. Como o indivíduo C4 tem o maior *fitness* da população, ele ocupa uma “fatia” maior na roleta, tendo assim uma maior probabilidade de ser selecionado. Suponha a seleção de um indivíduo. A roleta é “girada” e uma esfera é “jogada” na mesma. Supondo que a esfera tenha parado na casa 32, o indivíduo selecionado é o C3. O número de vezes que a roleta é girada corresponde ao número de indivíduos que se queira selecionar.

Na seleção por torneio, n indivíduos são escolhidos, aleatoriamente, com a mesma probabilidade.

Indivíduos	C1	C2	C3	C4	C5	C6	C7	C8
Fitness	12	13	19	25	08	06	08	09
Fitness acumulado	12	25	44	69	77	83	91	100

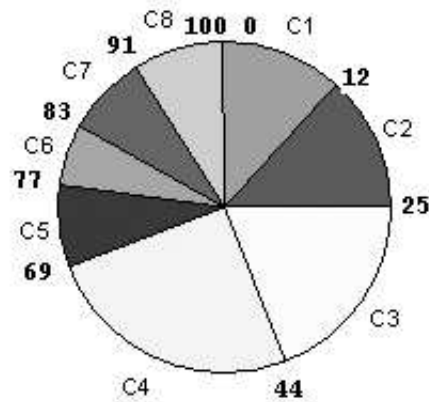


Fig. 2.5: Exemplo de método da roleta para selecionar indivíduos de uma população

O indivíduo com maior aptidão, entre os escolhidos, é selecionado para compor a população intermediária. O processo se repete até que se consiga a população intermediária com o número de indivíduos desejados. Este método de seleção pode ser implementado de forma simples e rápida porque nenhuma ordenação da população é necessária.

Na seleção por truncamento, apenas um subconjunto dos melhores indivíduos são escolhidos para uma possível seleção, com a mesma probabilidade. Este procedimento é repetido até que se consiga uma população intermediária com o número de indivíduos desejados. Esse método também é simples, mas como é necessária uma ordenação da população, ele tem uma maior complexidade de tempo quando comparado à seleção por torneio.

2.7.5 Reprodução

O processo de reprodução nos AGs acontece, basicamente, a partir da aplicação de dois operadores genéticos: o *crossover* (ou recombinação) e a *mutação*. Este processo ocorre no nível cromossômico, provocando alterações no mesmo.

Crossover ou Recombinação

Indivíduos selecionados são recombinaados através do operador de *crossover*. Este processo é inspirado na recombinação biológica sexuada, ou seja, pares de genitores são escolhidos através do processo de seleção e novos indivíduos são criados a partir da troca de material genético entre os pares. Os filhos serão diferentes de seus pais, mas com características genéticas de ambos os genitores.

O *crossover* é um operador muito discutido devido à sua natureza rompedora (isto é, ele pode separar informações importantes) [21]. De fato, outros algoritmos evolutivos não utilizam este operador, nem outro tipo similar de recombinação. Entretanto, nos AGs, a sua aplicação é fundamental com o intuito de preservar um bom material genético e possibilitar a formação de outros ainda melhores. A frequência com que o *crossover* ocorre é controlada pelo parâmetro p_c , probabilidade de recombinação ou taxa de *crossover*, que diz a porcentagem de população que a cada geração será submetida ao *crossover*.

As abordagens mais conhecidas e utilizadas para o *crossover* são: *crossover* ponto-simples (ou um-ponto), *crossover* múltiplo e *crossover* uniforme.

No *crossover* ponto-simples, os dois pais são “cortados” em um ponto específico, escolhido aleatoriamente, e os segmentos localizados depois do corte são trocados. Ou seja, os pais dão origem a dois filhos, onde o primeiro filho repete os genes do pai 1 até o ponto de *crossover* e, a partir deste ponto, ele repete os genes do pai 2. O segundo filho repete os genes do pai 2 até o ponto de *crossover* e, a partir deste ponto, ele repete os genes do pai 1. A Figura 2.6 ilustra o *crossover* de um-ponto.

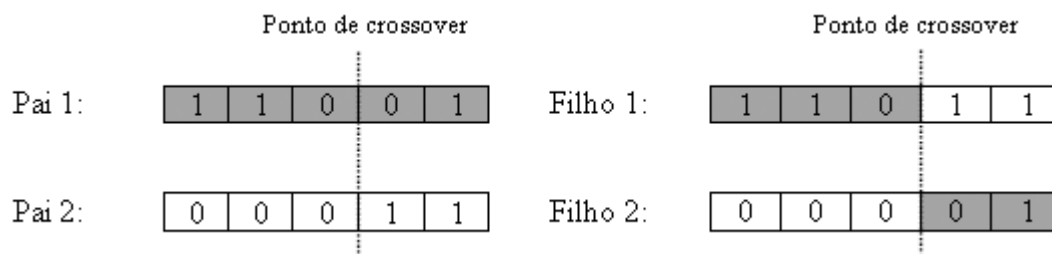
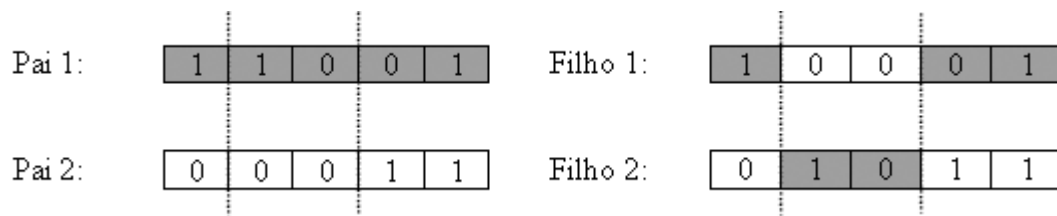


Fig. 2.6: Exemplo de *crossover* ponto-simples.

O *crossover* múltiplo é similar ao ponto-simples, com a diferença que são sorteados mais de um ponto de *crossover* e a troca de material genético é feita nos vários pontos. A Figura 2.7 ilustra como é feita a recombinação genética através de um *crossover* múltiplo com dois pontos de corte. Neste

Fig. 2.7: Exemplo de *crossover* dois-pontos.

tipo de *crossover*, os elementos entre os dois pontos sorteados são trocados entre os dois pais para formar os dois filhos.

O *crossover* uniforme é outro tipo importante de mecanismo de recombinação [105]. Nele, ao invés de sortear-se pontos de *crossover*, sorteia-se uma máscara do tamanho do cromossomo que indica qual cromossomo pai fornecerá cada gene para o filho 1. O filho 2 é gerado pelo complemento da máscara. A Figura 2.8 ilustra como é realizado o *crossover* uniforme. Este tipo de *crossover* é mais “rompedor” que o *crossover* dois-pontos. Por outro lado, o *crossover* dois-pontos é ineficiente quando a população converge rapidamente devido à incapacidade de promover a diversidade.

Fig. 2.8: Exemplo de *crossover* uniforme.

Mutação

A mutação é um operador equivalente a uma busca do tipo aleatória, cujo objetivo é aumentar a diversidade na população. Através dele, novos indivíduos são criados modificando-se um ou mais genes de indivíduos já existentes na população para outros valores (alelos) possíveis. São aleatórias as escolhas dos indivíduos que sofrerão mutação, dos genes e dos novos valores dos genes. No caso mais simples (Figura 2.9), onde a representação de um indivíduo é binária, a mutação de um gene é feita pelo complemento do valor atual, isto é, se o valor for igual a 0 (zero) será alterado para 1 e

vice-versa. Outro tipo de mutação muito conhecido é a *Permutação* (Figura 2.10). Para este tipo, há a troca de dois (ou mais) genes.

O processo de mutação é controlado por um parâmetro fixo p_m , que indica a probabilidade de um gene sofrer mutação. A mutação garante que a probabilidade de pesquisa em qualquer região do espaço nunca seja zero e previne a perda de material genético durante a seleção [27].



Fig. 2.9: Exemplo de mutação pelo complemento do alelo



Fig. 2.10: Exemplo de mutação por permutação

2.7.6 Reinscrição da população

Após a geração de indivíduos filhos, feita a partir da aplicação dos operadores de seleção, *crossover* e mutação, o AG faz uso do método de *reinscrição* para formar sua próxima população a partir da atual, considerando-se também os filhos gerados. Alguns métodos de reinscrição conhecidos são:

Reinscrição pura: todos os indivíduos selecionados para a reprodução (pais) serão substituídos por indivíduos filhos.

Reinscrição uniforme: alguns indivíduos selecionados para a reprodução serão escolhidos, de maneira aleatória e uniforme, para deixar a população, dando lugar aos indivíduos filhos.

Reinserção elitista: os indivíduos pais menos adaptados serão substituídos pelos filhos, e a elite (melhores indivíduos da população de pais) é mantida para a próxima geração.

Reinserção baseada na aptidão: a quantidade de indivíduos filhos é maior que o necessário para reinserção, sendo apenas os melhores filhos reinseridos na população.

O método de reinserção mais simples é a reinserção pura, onde cada indivíduo “vive” somente uma geração. No entanto, com a reinserção pura, bons indivíduos podem ser substituídos sem produzir filhos melhores. Logo, boas informações podem ser perdidas.

Os métodos de reinserção elitista e baseado na aptidão previnem esta perda de informação e por isso são mais recomendados. No elitismo, n indivíduos da população atual são copiados, sem nenhuma modificação, para a próxima geração. Isto geralmente resulta em uma busca mais “agressiva”, o que pode levar a bons resultados. No entanto, existe o perigo da convergência prematura (Seção 2.8) para mínimos locais. A seleção por elitismo pode ser usada em combinação com outras estratégias de seleção. O método baseado na aptidão implementa uma seleção por truncamento entre os filhos, para decidir quem são os melhores, antes de inseri-los na população.

Indivíduos pais podem ser substituídos por filhos que possuem um menor valor de aptidão. Com isso, a aptidão média da população pode diminuir. Entretanto, se os filhos inseridos possuem um valor de aptidão ruim, espera-se que eles sejam substituídos, na próxima geração, por novos filhos.

2.7.7 Critérios de Parada

O critério ideal para o término de um algoritmo genético seria assim que o ponto ótimo global fosse encontrado. Porém, na maioria das vezes não é possível afirmar se um ponto ótimo alcançado trata-se do ótimo global. Diante deste fato, pode-se utilizar quatro tipos de critérios de parada [27]:

- Encontrado um indivíduo com aptidão maior do que um limiar pré-estabelecido;
- Excedido um limite máximo de iterações, ou gerações, previamente estabelecido;
- Excedido o tempo máximo de processamento pré-estabelecido;
- Estagnação da população, quando após um número de iterações previamente estabelecido, o algoritmo não consegue melhorar a aptidão do melhor indivíduo da população.

2.8 Problemas dos Algoritmos Genéticos

Existem alguns problemas que são observados quando da utilização de algoritmos genéticos. Dentre eles, os mais comuns são: a convergência prematura, o tempo de processamento elevado e a dificuldade de definição dos parâmetros.

Convergência Prematura

A convergência prematura é um problema que está diretamente relacionado com a pressão seletiva e com a perda da diversidade da população onde, em poucas gerações, um AG pode convergir para um ponto de alta qualidade, mas que não é necessariamente o ótimo global. Ou seja, este problema acontece quando surge um indivíduo na população que muito rapidamente vem a dominá-la, levando o AG a convergir para um máximo local. Suponhamos que, na população inicial, haja um indivíduo próximo de um ótimo local, mas não global, que tenha um valor de aptidão muito maior que o restante da população. O processo de seleção adotado pode diversas vezes selecionar este indivíduo para reprodução, gerando vários descendentes deste indivíduo nas gerações seguintes. Em casos extremos, indivíduos próximos ao ótimo global mas com baixa aptidão relativa, podem ser completamente extintos, o que geralmente resulta em convergência na população para um ótimo local [106, 107]. Uma vez que a população convergiu, a habilidade do AG para continuar a busca por melhores soluções é quase eliminada. Neste caso, somente a mutação, com seu mecanismo de perturbação aleatória, pode explorar novas regiões do espaço de busca [21].

O conceito de diversidade populacional é abstrato, e indica o grau em que as mais diversas regiões do espaço de busca estão representadas na população. É importante que, na população inicial, esta diversidade seja grande, explorando ao máximo as diversas regiões do espaço de busca. Em [106], foram citadas duas maneiras de se promover o aumento da diversidade populacional: aumentando-se a taxa de mutação (p_m) ou o tamanho da população.

Outro fator que pode levar à convergência prematura é a pressão seletiva. Esta pressão, que depende da função de avaliação e do método de seleção adotado, é que determina a direção e a velocidade que o AG vai explorar o espaço de busca. Aumentando-se a pressão seletiva, a probabilidade de alguns indivíduos da população serem selecionados torna-se maior em relação a outros indivíduos. À medida que a pressão seletiva vai se tornando menor, as probabilidades de sobrevivência vão

sendo melhor distribuídas entre os indivíduos.

Permitir que apenas um pequeno grupo de indivíduos possa sobreviver pode causar uma perda da diversidade da população. Assim, aumentando-se a pressão seletiva perde-se a diversidade populacional e acelera-se a convergência do AG para algum ponto de ótimo, local ou global. Se os indivíduos favorecidos pela pressão seletiva estiverem próximos de um máximo global, ou seja, se forem um resultado satisfatório, a pressão seletiva estará atuando como um fator benéfico, pois o tempo de processamento será reduzido. Caso contrário, se os indivíduos favorecidos não representarem resultados satisfatórios, a pressão seletiva estará atuando como um fator prejudicial, podendo levar à convergência prematura.

A idéia básica para prevenir a convergência prematura, é controlar o número de oportunidades de reprodução de cada indivíduo. A estratégia é comprimir a taxa de adaptabilidade da população, sem perder a pressão seletiva, e prevenir que qualquer super-indivíduo possa inesperadamente dominar a população [21]. Ou seja, deve haver um equilíbrio entre a pressão seletiva e a diversidade populacional.

Tempo de Processamento Elevado

A maioria das aplicações de AGs é para problemas considerados complexos. Sendo assim, o espaço de busca pode ser muito grande e/ou ser lenta a avaliação de todos os indivíduos da população. Neste caso, a complexidade do problema pode fazer com que o AG apresente um tempo de processamento elevado.

Uma excelente solução para esta questão é a utilização de processadores paralelos na execução de AGs [27]. Como a natureza dos AGs é intrinsecamente paralela, esta é uma abordagem que tem dado bons resultados. Por exemplo, como a medida da aptidão de um indivíduo é independente da medida do outro, pode-se pensar em uma rede de processadores em paralelo, onde cada processador é responsável por medir a aptidão de um indivíduo, acelerando a computação de uma geração no AG.

Dificuldade de Definição dos Parâmetros

Ao optar pela utilização de algoritmos genéticos, o “projetista” do algoritmo depara-se com um problema: a escolha dos parâmetros. Estes parâmetros, assim com a representação do problema e a

função de avaliação (aptidão), serão dependentes do problema a ser resolvido. Para que o AG tenha o comportamento esperado e possa alcançar soluções satisfatórias, é de fundamental importância que tais parâmetros sejam bem definidos.

Nos AGs considerados mais simples, os parâmetros numéricos que diretamente influenciam no comportamento do AG são três: tamanho da população, probabilidade de *crossover* (p_c) e probabilidade de mutação (p_m).

Há alguns resultados empíricos que têm mostrado que bons resultados são geralmente obtidos com os valores da Tabela 2.1 [7, 18, 107]:

Pesquisador	Tamanho População	Prob. <i>Crossover</i> (%)	Prob. Mutação (%)
De Jong	50 - 100	60	0,01
Grefenstette	30	95	1
Schaffer	20 - 30	75 - 95	0,5 - 1
Goldberg	50 - 200	≥ 70	≤ 1

Tab. 2.1: Determinação de parâmetros de AGs, segundo [7, 18, 107]

2.9 Algumas Aplicações de Algoritmos Genéticos

Os algoritmos genéticos têm sido aplicados, com sucesso, na resolução de vários tipos problemas. Dentre eles estão [106]:

Otimização: otimização numérica e problemas de otimização combinatorial como *layout* de circuitos, escalonamento *job-shop* e escalonamento em multiprocessadores [53].

Programação automática: evolução de programas de computador para tarefas específicas e projeto de outras estruturas computacionais como os autômatos celulares [34];

Aprendizado de máquina: tarefas de classificação e previsão, como a previsão do clima ou a estrutura de proteínas [59];

Sistemas econômicos: modelar processos de inovação, estratégias de concorrência e licitação [50];

Sistema de imunização: modelar vários aspectos de sistemas de imunização natural, incluindo mutação somática e tempo de vida dos indivíduos [10];

Ecologia: modelar fenômenos ecológicos, como por exemplo, coevolução hospedeiro-parasita [48];

Robótica: determinação da trajetória do movimento de braços mecânicos, projeto e controle de robôs [24];

Mineração de dados: descobrir informações relevantes, como padrões, associações, mudanças, anomalias e estruturas, em grandes quantidades de dados armazenados em banco de dados [18];

Redução de perdas: reduzir perdas em redes de distribuição de energia elétrica [110].

2.10 Algoritmo Genético Coevolutivo

Os algoritmos genéticos coevolutivos (AGC) [48] diferem do algoritmo genético tradicional, também conhecido como AG padrão, por utilizarem duas populações durante o processo evolutivo: uma população de possíveis soluções e outra população de instâncias do problema a resolver.

Os métodos coevolutivos, por sua vez, podem ser classificados em competitivos e cooperativos. Embora a maioria dos AGCs implementados baseiam-se em modelos com as espécies em competição, trabalhos recentes exploram o uso de modelos com espécies em colaboração [79, 83, 84]. Nos casos cooperativos, a aptidão do indivíduo de uma população é determinada por uma série de *colaborações* com indivíduos da outra população [84]. Neste caso, tem-se um tipo de simbiose, onde o reforço de aptidão entre as populações é positivo. Nos casos competitivos, a aptidão do indivíduo de uma população é determinada por um série de *competições* com indivíduos da outra população [81]. Assim, a interação de aptidão é negativa, e o sucesso de um indivíduo é o fracasso do outro, seguindo a metáfora “presa-predador”.

A maioria dos AGCs implementados baseia-se na metáfora presa-predador. Na natureza, a interação presa-predador provê, nas espécies envolvidas, uma evolução muito forte. Isto acontece porque a pressão evolutiva sobre a presa para se defender (por exemplo, camuflagem) acontecerá como resposta à evolução da estratégia de ataque de um predador nas gerações futuras (por exemplo, melhoria de visão), e vice-versa, a melhoria da capacidade de defesa acarretará uma melhoria na estratégia de ataque. Esta luta pela sobrevivência é caracterizada como uma função de avaliação inversa, onde o sucesso de um dos lados é caracterizado pelo fracasso do outro lado, que deve ser responder, através

da evolução, para manter a chance de sobrevivência.

Foi através do trabalho de Hillis [48] que percebeu-se que o conceito de coevolução poderia ser utilizado na área de computação. Ele utilizou a coevolução (considerando a metáfora presa-predador) para achar uma arquitetura de rede ótima dentro de uma lista de arquiteturas possíveis. A partir destes experimentos, Hillis introduziu o conceito de Algoritmo Genético Coevolutivo (AGC).

O AGC baseado na metáfora presa-predador manipula duas populações distintas e que interagem entre si: uma população é formada por indivíduos (cromossomos) que representam possíveis soluções para o problema e a outra é formada por instâncias deste problema, ou seja, testes utilizados para avaliar a qualidade de uma solução. A Figura 2.11 apresenta um esquema comparativo da estrutura das populações no AG padrão e no AGC baseado na metáfora presa-predador.

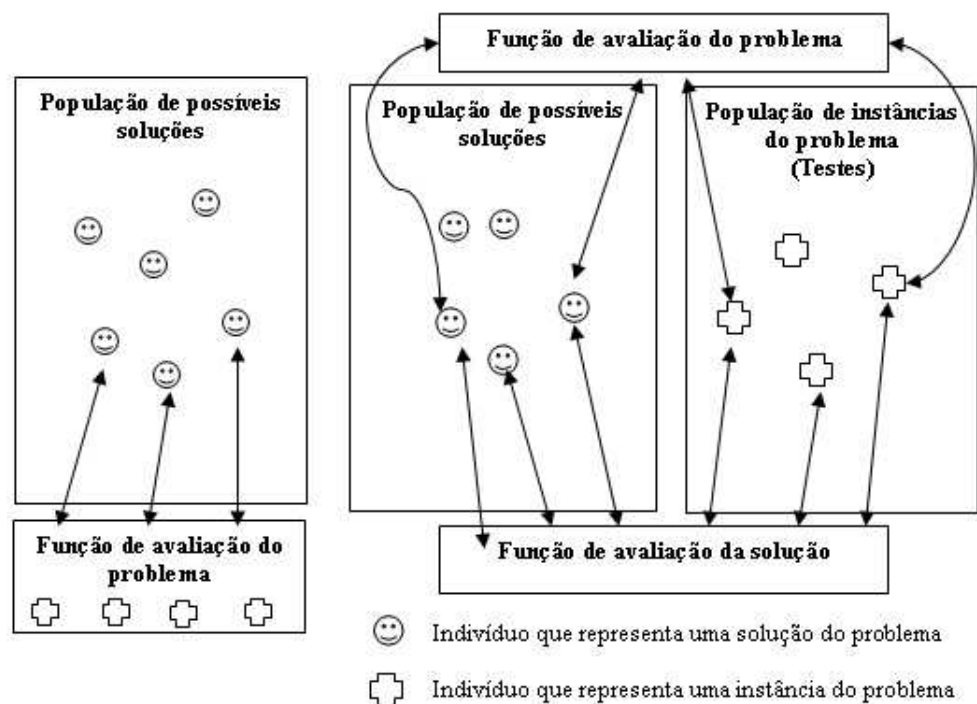


Fig. 2.11: (a) Estruturas das populações nos AGs e (b) nos AGCs.

Vários trabalhos apresentaram bons resultados obtidos com o uso dos AGCs. Dentre eles, podemos citar: encontrar uma melhor arquitetura para uma Rede Neural [78], problemas de satisfação de restrições [77], Tarefa da Classificação da Densidade [81] e Evolução de Autômatos Celulares utilizando Algoritmo Genético Coevolutivo [56]. Os AGCs foram, em relação aos AGs tradicionais, mais eficazes na resolução de alguns problemas [56, 81]. Os pesquisadores atribuíram esta melhoria devido

ao fato dos AGCs evitarem uma estagnação precoce da população de possíveis soluções, diminuindo o efeito da convergência prematura nesta população.

Quando, no AGC, as populações de problemas e soluções são “evoluídas” ao longo das gerações com a utilização de operadores genéticos (*crossover* e *mutação*), é dito que o AGC é do tipo total [81, 114]. Caso contrário, quando apenas a população de soluções é evoluída, é dito que o AGC é do tipo parcial [56, 80].

2.11 Ferramentas estatísticas aplicadas na análise de dados obtidos pelos AGs

Quando os AGs são aplicados na resolução de vários problemas complexos, são realizados estudos *experimentais* ou *observacionais*, o que leva a uma coleção de *dados* numéricos (por exemplo, médias, desvio padrão, etc.). Em muitos casos, a interpretação dos dados (resultados) obtidos através de um experimento não é uma tarefa trivial, pois o padrão de variação nestes dados não deixa óbvio se eles são realmente bons ou ruins.

Alguns métodos estatísticos podem ser utilizados na interpretação dos dados obtidos pelos AGs. Dentre eles, podemos citar o Intervalo de Confiança e o Teste de Hipótese [73], os quais foram utilizados, nos Capítulos 5 e 6, para interpretar e avaliar os dados obtidos através de resultados experimentais.

O intervalo de confiança para um **parâmetro populacional verdadeiro** (mas desconhecido) fornece um intervalo de valores plausíveis para este parâmetro baseado nos dados amostrais. Sendo assim, um intervalo de confiança de $N\%$ para um parâmetro populacional fornece um intervalo no qual estaríamos $N\%$ confiantes de cobertura do verdadeiro valor do parâmetro. Por exemplo, podemos implementar um AG para a minimização de uma função $y = f(x)$ qualquer. Ao executar esse AG a partir de diferentes populações iniciais, diferentes valores de y podem ser obtidos. Suponha que em k execuções desse AG a média dos valores de y obtidos ao final de cada execução foi $\bar{y}$, com um desvio padrão σ associado a essa média. O intervalo de confiança é uma ferramenta estatística que nos permite dizer, por exemplo, que estamos 95% confiantes que ao executar o AG, o valor de y a ser obtido estará dentro de um intervalo $[y_{min}, y_{max}]$, sendo que os valores de y_{min} e y_{max} dependem dos

valores de $\bar{y}$, σ e k .

Os testes de hipóteses têm como objetivo decidir, com base na informação fornecida pelos dados de uma amostra, sobre a aceitação ou não de uma dada hipótese. Intervalos de confiança e testes de hipóteses estão intimamente relacionados. Em geral, intervalos de confiança são a forma mais informativa de apresentar os achados principais de um estudo. Contudo, algumas vezes existe um particular interesse em decidir sobre a verdade ou não de uma hipótese específica (se dois grupos têm a mesma média ou não, ou se o parâmetro populacional tem um valor em particular ou não), e o teste de hipóteses fornece-nos a estrutura para que façamos isto. Por exemplo, se fizermos duas implementações diferentes de AG para minimizar a mesma função $y = f(x)$ e executarmos cada AG implementado k vezes, obteremos dois intervalos de confiança diferentes, cada um associado a uma das implementações. Com base nesses intervalos, podemos aplicar o teste de hipóteses para definirmos, por exemplo, que estamos 90% confiantes que uma execução da implementação 1 obterá um valor de y inferior ao obtido por uma execução da implementação 2.

Uma explicação mais detalhada sobre os intervalos de confiança e os testes de hipótese é apresentada no Apêndice A.

Capítulo 3

Autômatos Celulares

Este capítulo refere-se aos Autômatos Celulares (ACs), sistemas dinâmicos discretos de simples implementação, porém, capazes de fazer emergir um comportamento complexo. Na Seção 3.1 é apresentada uma breve introdução aos ACs, suas origens, motivações e principais pesquisadores. Na Seção 3.2 são apresentados os principais conceitos, definições e notações utilizadas. Na Seção 3.3 são apresentados os ACs unidimensionais e bidimensionais. Na Seção 3.4 é discutida a dinâmica dos ACs e a utilização de alguns parâmetros na previsão do comportamento dos mesmos. A Seção 3.5 discute a computação baseada em ACs. A Seção 3.6 apresenta a resolução da Tarefa de Classificação da Densidade (TCD) através da busca evolutiva de regras de ACs.

3.1 ACs: história, origens e motivação

Os autômatos celulares (ACs) são sistemas dinâmicos discretos (variáveis, tempo e espaço) formados por componentes simples e idênticos (conhecidos como *células*) com conectividade local. Estes sistemas, além de serem de implementação simples, possuem uma série de parâmetros que podem ser diretamente manipulados, facilitando o estudo de sua dinâmica e incentivando a sua utilização, em diferentes áreas, na modelagem de diversos sistemas complexos reais [47, 118, 123]. Muitos pesquisadores observaram que, apesar da simplicidade de implementação, os ACs apresentam comportamentos dinâmicos interessantes e complexos.

O autômato celular mais simples é formado por uma “linha” de células (o reticulado) onde cada

célula pode assumir o valor 0 ou 1. As células interagem localmente em um tempo discreto t , usualmente de forma paralela (síncrona). O AC também é caracterizado por uma regra de transição de estados, que determina qual será o próximo estado do reticulado, a partir de seu estado atual.

Os autômatos celulares foram originalmente propostos por John von Neumann, na década de 50, como modelos formais de organismos com a capacidade de auto-reprodução. As estruturas estudadas eram, na maioria das vezes, grades unidimensionais e bidimensionais, embora dimensões maiores também fossem consideradas. O interesse de von Neumann na época era identificar um tipo de organização lógica para um autômato que fosse capaz de fazê-lo se auto-reproduzir. Neste contexto, ele prôpos-se a encontrar uma máquina de Turing capaz da sua auto-reprodução [74]. Paralelo aos trabalhos de von Neumann, outro pesquisador, Stannislav Ulam, estudava vários jogos matemáticos dispostos em uma grade bidimensional onde o estado de cada ponto (célula) na grade era atualizado de acordo com o seu próprio estado e os estados de seus pontos vizinhos. O estado de um determinado ponto, juntamente com os estados de seus vizinhos, formavam a vizinhança deste ponto. Ulam estudava diferentes maneiras de formação da vizinhança, com diversos números de estados possíveis por células e regras de transição variadas.

Através de seus estudos, Ulam sugeriu a von Neumann o conceito de uma grade artificial, semelhante a um tabuleiro de xadrez, na qual cada quadrado (casa) poderia ser visto como uma célula (ou ponto), e cada célula apresentando um comportamento individual, baseado em um conjunto de regras, que seria aplicado a todas as células da grade individualmente. A atualização do estado das células seria realizada em passos discretos de tempo. Cada célula da grade “olharia” o seu próprio estado e o estado de suas células vizinhas e, a cada passo de tempo, um conjunto de regras de transição seria consultado para decidir qual o próximo estado de cada célula da grade. Um conjunto de células nesta grade poderia ser considerado como um organismo artificial. Seguindo a sugestão de Ulam, von Neumann construiu o primeiro modelo de autômato celular, que resolvia a questão da auto-reprodução. Neste modelo, que encapsulava uma máquina universal de Turing, a vizinhança de cada célula era formada por 5 células e cada célula podia assumir um de 29 estados possíveis. Os resultados encontrados através da aplicação deste AC foram publicados após a morte de von Neumann, por seu colaborador Arthur Burks [74]. Anos depois, vários pesquisadores deram continuidade a esta pesquisa, encontrando ACs menos complexos, que também eram capazes de se auto-reproduzir

[20, 28].

John Conway [9], outro pesquisador que merece destaque na história dos ACs, no início da década de 70, estudava a possibilidade de criar um autômato celular simples com computabilidade universal. O resultado de seus estudos foi o *Game of Life* (ou simplesmente *Life*), um AC bidimensional binário (cada célula só pode assumir o valor 0 ou 1) com uma regra de transição bastante simples, onde células no estado 0 são interpretadas como “mortas” e as no estado 1 como “vivas”. Através do *Life*, Conway popularizou o AC nos meios acadêmicos, pois este modelo de AC, apesar de ser relativamente simples, mostrou sua habilidade em produzir padrões complexos e estruturas que se assemelhavam a organismos artificiais.

Na década de 80, destacam-se os estudos feitos sobre o comportamento dinâmico do autômatos celulares [118, 119, 120, 121, 123, 124]. Estes estudos, liderados por Stephen Wolfram, tornaram-se fonte de referência de todos os trabalhos posteriores nesta área. Uma das principais contribuições de Wolfram, que mudou o rumo das pesquisas na área, foi demonstrar que mesmo os modelos de ACs mais simples (unidimensionais, binários e vizinhança formada por 3 células) seriam capazes de exibir padrões interessantes, com um comportamento emergente. Recentemente, Wolfram provou que esses mesmos ACs possuem computabilidade universal [125].

Devido à sua simplicidade de implementação e dinâmica complexa, pesquisadores têm utilizado os autômatos celulares para modelar fenômenos reais, derivando importantes resultados dos modelos obtidos. Alguns exemplos de modelamentos através de ACs são: proliferação de epidemias [38], simulação de fenômenos físicos [111], simulação de tráfego urbano [115] e formação de padrões em conchas e peles de animais [67, 125].

Vida Artificial (VA) é um ramo relativamente recente da computação que estuda sistemas criados artificialmente que incorporem pelo menos alguns componentes característicos da vida real [62]. Atualmente, os ACs são muito utilizados em VA [62, 64, 108], por se tratarem de sistemas cuja dinâmica permite uma analogia com a dos sistemas vivos.

Dentre as habilidades dos ACs que mais atraem o interesse dos pesquisadores, está a sua capacidade de realizar computações baseadas em interações puramente locais. Por se tratarem de estruturas paralelas, com uma lógica simples, os ACs têm sido cogitados como uma opção para arquiteturas descentralizadas de computadores. Porém, o entendimento de como o AC consegue realizar computação

é algo ainda não trivial e, na busca de ACs com habilidades computacionais específicas, a utilização de técnicas evolutivas tem sido uma abordagem bem sucedida [70].

Além do modelamento de fenômenos reais, os ACs têm sido utilizados na resolução de tarefas computacionais pré-definidas, tais como, a tarefa de Classificação da Densidade (TCD) [69, 71, 72], a tarefa de Sincronização [22], o escalonamento de tarefas em sistemas multiprocessadores [93, 95, 96, 97, 102, 104] e a criptografia [30, 47, 122].

3.2 Conceitos, Definições e Notação

Os ACs são sistemas compostos por dois componentes: o Espaço Celular e a Regra de Transição. O espaço celular é um reticulado de N células idênticas dispostas em um arranjo d -dimensional, cada uma com um padrão idêntico de conexões locais para outras células, e com condições de contorno. A condição de contorno de um reticulado pode ser periódica ou não-periódica. Quando a condição de contorno é periódica, as células da extremidade do reticulado estão conectadas, em uma espécie de anel. Ao contrário, quando a condição de contorno é não-periódica, as células da extremidade não estão conectadas, sendo necessário definir a forma de atualização das bordas. A regra de transição é uma função que retorna o próximo estado para cada célula, levando-se em consideração o seu estado atual e o de suas células vizinhas (vizinhança). A cada passo de tempo discreto, todas as células do reticulado têm o seu estado atualizado de acordo com esta regra. O conjunto de valores (estados) destas células em um determinado instante de tempo t representa uma **configuração** do AC naquele instante. Ou seja, no instante t_0 , tem-se a **configuração inicial** do reticulado do AC.

A notação aqui adotada para definir os ACs e seus componentes foi a mesma adotada em [68]. O conjunto dos possíveis estados (valores) que cada célula pode assumir é denotado por Σ e o número de elementos deste conjunto é denotado por k . Cada célula do reticulado é denotada por um índice i e seu estado a um dado tempo t é denotado por S_i^t , onde $S_i^t \in \Sigma$. A vizinhança de uma célula i , denotada por η_i^t , é formada pelo estado S_i^t da célula i e pelos estados das células às quais a célula i está localmente conectada. A regra (função) de transição é denotada por $\Phi(\eta_i)$ que fornece o próximo estado S_i^{t+1} para cada célula i , como uma função de η_i^t . Ou seja, o estado de uma dada célula i em um instante de tempo $(t + 1)$ é definido em função de sua vizinhança em um instante de tempo t . A

cada passo de tempo, todas as células do reticulado do AC atualizam seu estado, usualmente de forma síncrona, de acordo com $\Phi(\eta_i)$.

A Figura 3.1 apresenta um AC unidimensional binário ($k = 2$ e $\Sigma = \{0,1\}$), com um reticulado formado por 10 células ($N = 10$). A vizinhança η_i de cada célula i que compõe o reticulado é formada por ela mesma e mais duas células vizinhas: $i - 1$ (célula à esquerda de i) e $i + 1$ (célula à direita de i). As condições de contorno são periódicas, ou seja, a célula da extremidade esquerda do reticulado (célula 0) está conectada à célula da extremidade direita (célula 9), e vice-versa. Uma configuração inicial do reticulado é dada pelos estados das células no tempo inicial ($t = 0$). Nos ACs binários, a regra de transição $\Phi(\eta_i)$ é dada por uma tabela de transição, onde para cada vizinhança possível, é mapeado um *bit* (valor) de saída, que representa a atualização do estado da célula central da vizinhança. A Figura 3.1 também apresenta a regra de transição formada pelas oito vizinhanças possíveis, considerando-se uma vizinhança de três células, e seus *bits* de saída (atualização). Por exemplo, como em $t = 0$ a vizinhança da célula 4 é 100, em $t = 1$ seu estado será 0, pois a regra de transição estabelece que se a vizinhança de uma determinada célula é 100, o seu próximo estado deverá ser mapeado para 0. Após realizada esta atualização para todas as células do reticulado, de forma síncrona (isto é, todas as células atualizam o seu estado ao mesmo tempo), obtemos no tempo $t = 1$ o novo reticulado apresentado na Figura 3.1c, que também demonstra a atualização dos estados das células do reticulado, ao se aplicar a regra de transição por 2 passos de tempo. Esta atualização ao longo do tempo é também chamada de *evolução temporal*.

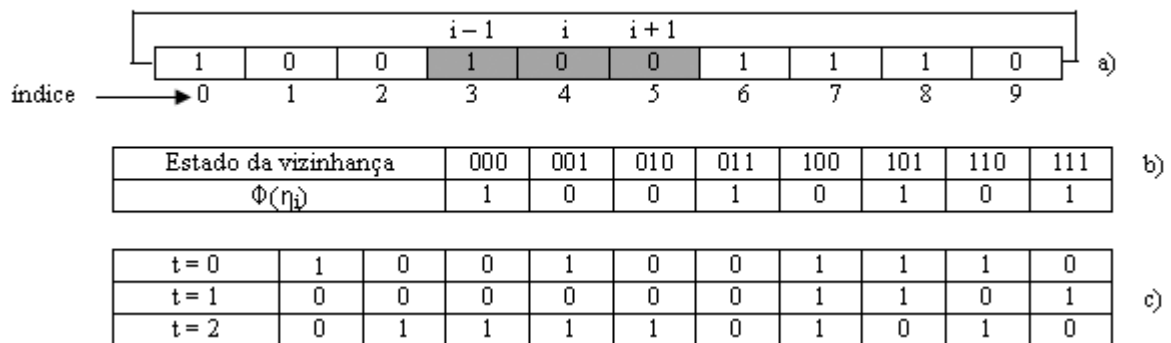


Fig. 3.1: Exemplo de um AC unidimensional binário com condição de contorno periódica: (a) reticulado, (b) regra e (c) evolução temporal.

3.2.1 Condições de Contorno

Como mencionado anteriormente, são duas as possíveis condições de contorno para as células de um autômato celular. Na condição de contorno *periódica* as células da extremidade do reticulado estão conectadas, como se o reticulado fosse um “anel”. A Figura 3.1a apresenta um reticulado com condição de contorno periódica. Em $t = 0$, a vizinhança da célula 0 é 010 (formada pelo estado das células 9, 0 e 1) e a vizinhança da célula 9 é 101 (formada pelo estado das células 8, 9 e 0). Na condição de contorno *não-periódica* as células da extremidade do reticulado não estão conectadas. Neste caso, o modelo do AC deve prover meios para a formação da vizinhança das células da extremidade. No caso mais comum, as células “ausentes” na formação da vizinhança das células da extremidade são consideradas no estado 0. Estas células são utilizadas apenas para complementar a vizinhança, não sendo em momento algum atualizadas. A Figura 3.2 apresenta a evolução do reticulado da Figura 3.1, com a mesma regra de transição, mas com condição de contorno não-periódica. Em $t = 0$, a vizinhança da célula 0 é 010 (formada pelo estado da célula ausente e das células 0 e 1) e a vizinhança da célula 9 é 100 (formada pelo estado das células 8, 9 e a célula ausente).

0	1	0	0	1	0	0	1	1	1	0	0
0	1	2	3	4	5	6	7	8	9		

a)

Estado da vizinhança	000	001	010	011	100	101	110	111
$\Phi(\eta_i)$	1	0	0	1	0	1	0	1

b)

$t = 0$	1	0	0	1	0	0	1	1	1	0
$t = 1$	0	0	0	0	0	0	1	1	0	0
$t = 2$	1	1	1	1	1	0	1	0	0	1

c)

Fig. 3.2: Exemplo de um AC unidimensional binário com condição de contorno não-periódica: (a) reticulado, (b) regra e (c) evolução temporal.

3.2.2 Modos de Operação

Os ACs podem evoluir sob três modos de operação diferentes: paralelo (síncrono), seqüencial (as-síncrono) ou aleatório (seqüencial-aleatório). No modo de operação paralelo, o mais comum, todas as células do reticulado atualizam o seu estado ao mesmo tempo, isto é, as células são atualizadas sincronamente. No modo seqüencial [102], somente uma célula atualiza o seu estado em um dado

instante de tempo. A atualização inicia-se com a célula de índice 0, em seguida a célula 1 e continua progressivamente até a célula $N - 1$. Isto significa que, ao atualizar o seu estado, a célula i utiliza o estado já atualizado da célula $i - 1$. No modo aleatório, a atualização do estado das células é semelhante ao modo seqüencial, porém, a ordem em que ocorre esta atualização é aleatória. Ou seja, a atualização não necessariamente começa pela célula 0. Ela pode, por exemplo, começar pela célula 5, em seguida atualizar a célula 1, e assim sucessivamente, até que as N células do reticulado tenham sido atualizadas. Mas, a cada nova atualização, os estados das células já atualizadas são considerados.

3.3 Autômatos Celulares Unidimensionais e Bidimensionais

Embora os ACs possam ser definidos em qualquer dimensão, os mais usuais são os unidimensionais e os bidimensionais, os quais descrevemos a seguir.

3.3.1 Autômatos Celulares Unidimensionais

Os ACs unidimensionais são aqueles cujo reticulado é representado por uma “linha” de células. O tamanho da vizinhança η_i é usualmente dado pela fórmula $m = 2r + 1$, onde r é chamado o **raio do AC**. O raio indica quantas células à direita e à esquerda da célula central deverão ser consideradas para a formação da vizinhança. Por exemplo, seja $r = 1$. Neste caso, a vizinhança de uma dada célula i será formada pela célula $i - 1$, pela própria célula i e pela célula $i + 1$. Sendo assim, o tamanho da vizinhança é $m = 2 \times 1 + 1 = 3$. O cálculo é feito de forma similiar para raios maiores. A Figura 3.3a apresenta um AC unidimensional binário de raio $r = 1$ e a Figura 3.3b apresenta um AC unidimensional binário de raio $r = 2$, ambos formados por um reticulado de tamanho $N = 5$ e com condição de contorno não-periódica (o estado para as células ausentes na vizinhança é 0). As figuras destacam a vizinhança da célula $i = 4$.

Nos ACs unidimensionais tem-se a regra de transição $\Phi : \Sigma_{2r+1} \rightarrow \Sigma$. Sendo k o número de estados possíveis por célula, a fórmula utilizada para se calcular o tamanho da regra de transição é $n = k^{2r+1}$, e o espaço de regras, formado pelo conjunto todas as seqüências finitas possíveis com comprimento n , é dado pela fórmula k^n . Por exemplo, se $k = 2$ e $r = 2$, tem-se que o comprimento da regra é $n = 2^{(2 \times 2) + 1} = 32$ e que o espaço de regras é formado por um conjunto de 2^{32} diferentes

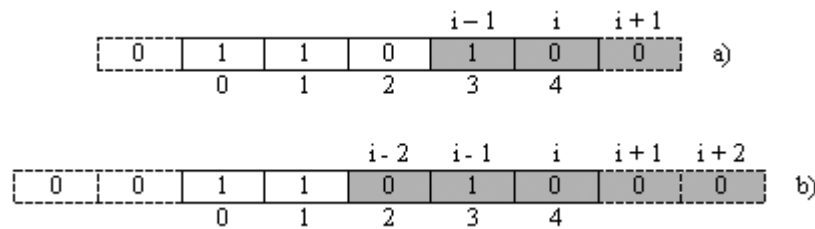


Fig. 3.3: Exemplos de AC unidimensional binário: (a) raio $r = 1$, (b) raio $r = 2$.

regras.

Conforme mencionamos anteriormente, o modelo de AC mais simples é o unidimensional, binário ($k = 2$) e de raio $r = 1$. Neste tipo de AC, tem-se então que $n = 8$ e o espaço de regras é composto por $2^8 = 256$ regras. Estes autômatos celulares são também chamados de *elementares*. Um exemplo de AC elementar é apresentado na Figura 3.1. Wolfram [118] propôs um esquema de numeração para os ACs elementares, no qual *bits* de saídas são ordenados lexicograficamente, como na regra de transição da Figura 3.1b, e são lidos da direita para a esquerda para formar um número na base decimal entre 0 e 255. Por exemplo, a regra da Figura 3.1b é o AC elementar 169. Para raios maiores que 1, o mais comum é representar a regra na base hexadecimal. Por exemplo, a regra de raio $r = 2$ dada pelos bits de saída 11101010110000111000101011101000 é equivalente à regra EAC38AE8.

Para observar o comportamento dos ACs unidimensionais, costuma-se utilizar os diagramas de padrões espaço-temporais. A Figura 3.1c apresenta um exemplo de diagrama espaço-temporal de um AC evoluído em 3 passos. O reticulado é mostrado horizontalmente e o tempo é incrementado verticalmente para baixo. Nestes diagramas, a configuração de estados em um reticulado é apresentada em função do tempo. Por exemplo, no tempo $t = 0$ tem-se a configuração inicial do reticulado. No tempo $t = 1$ tem-se uma nova configuração que resulta da aplicação da regra de transição no reticulado inicial. No tempo $t = 2$ tem-se uma nova configuração que resulta da aplicação da regra de transição no reticulado do tempo $t = 1$, e assim sucessivamente. Uma outra maneira de representação de diagramas espaço-temporais é apresentada na Figura 3.4, na qual as células no estado 0 são desenhadas na cor preta e aquelas no estado 1 são desenhadas na cor branca.

O espaço de regras cresce muito rápido, em função do tamanho n da regra. Por esta razão, alguns tipos de regras são usados com a finalidade de tornar as regras de transição dos ACs menores e, com isso, diminuir o espaço de regras. Estas regras são chamadas de *regras totalísticas*. Neste tipo de

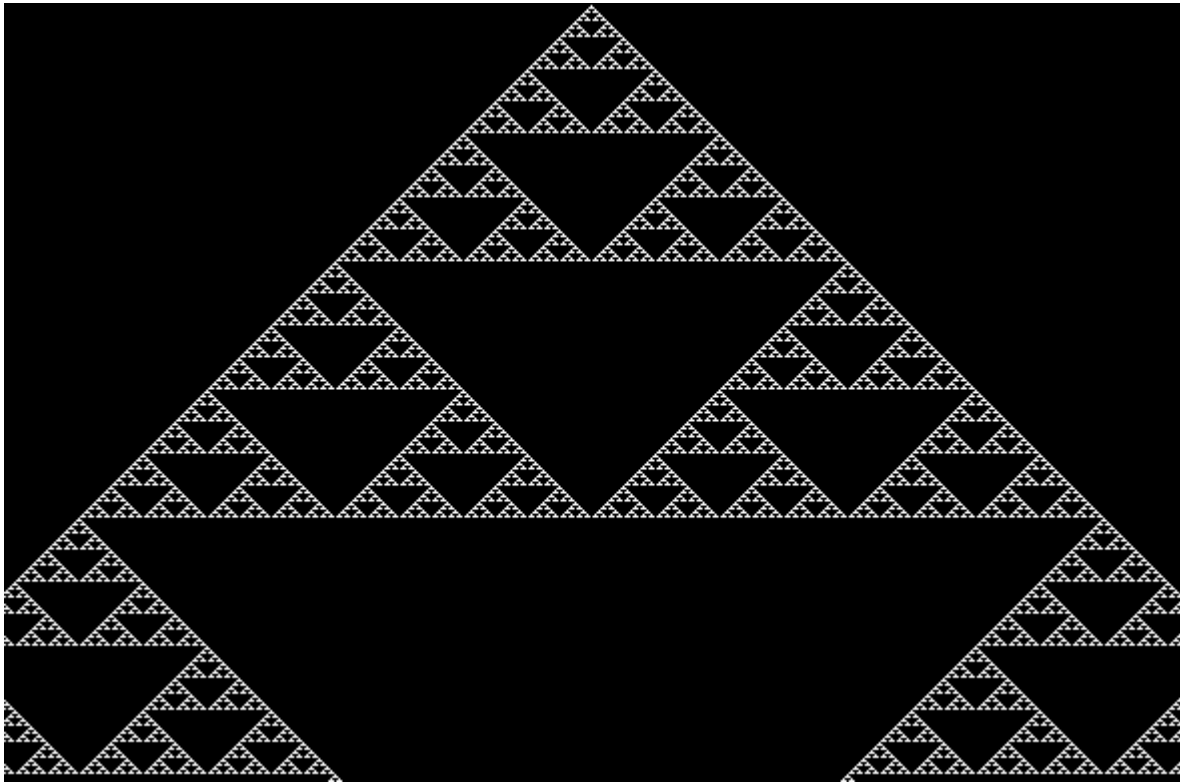


Fig. 3.4: Diagrama de padrões espaço-temporais de um AC [117].

regra de transição, a atualização dos estados das células é feita com base no somatório dos estados das células na vizinhança, isto é,

$$S_i^{t+1} = \Phi_t(S_{i-r}^t + \dots + S_{i-1}^t + S_i^t + S_{i+1}^t + \dots + S_{i+r}^t) \quad (3.1)$$

Por exemplo, para um AC com raio $r = 2$, o comprimento de uma regra de transição totalística é reduzido para $n = 6$, e o número total de regras no espaço de regras é 2^6 . Isto acontece porque, por exemplo, os estados da vizinhança 00001, 00010, 00100, 01000 e 10000 mapeiam todos para um mesmo estado da célula central, uma vez que o somatório dos bits da vizinhança é igual a 1. A Tabela 3.1 apresenta três exemplos de regras totalísticas para um AC unidimensional binário de raio $r = 2$.

Somatória dos Estados da Vizinhança:	0	1	2	3	4	5
Regra Φ_1	1	1	0	0	0	1
Regra Φ_2	0	1	1	1	0	1
Regra Φ_3	0	0	1	1	1	1

Tab. 3.1: Exemplos de regras totalísticas para um AC unidimensional binário de raio 2.

A complexidade dos ACs pode ser aumentada de várias formas [5]: alterando-se a dimensão do reticulado, o número de estados por célula ou o tamanho da vizinhança. Diferentes modelos de ACs também podem ser obtidos modificando-se as condições de contorno, tornando-se as regras de transição estocásticas ao invés de determinísticas, tornando-se a vizinhança não local [65], ou mesmo tornando-se as regras de transição não uniformes dentre as células [100]. A seguir, serão apresentados os ACs definidos em reticulados bidimensionais.

3.3.2 Autômatos Celulares Bidimensionais

Os ACs bidimensionais têm suas células arranjadas em um reticulado definido no plano bidimensional. A Figura 3.5a apresenta um exemplo de reticulado de um AC bidimensional binário. As vizinhanças bidimensionais mais utilizadas são apresentadas na Figura 3.5b e 3.5c: a vizinhança de von Neumann, composta por 5 células, e a vizinhança de Moore, composta por 9 células.

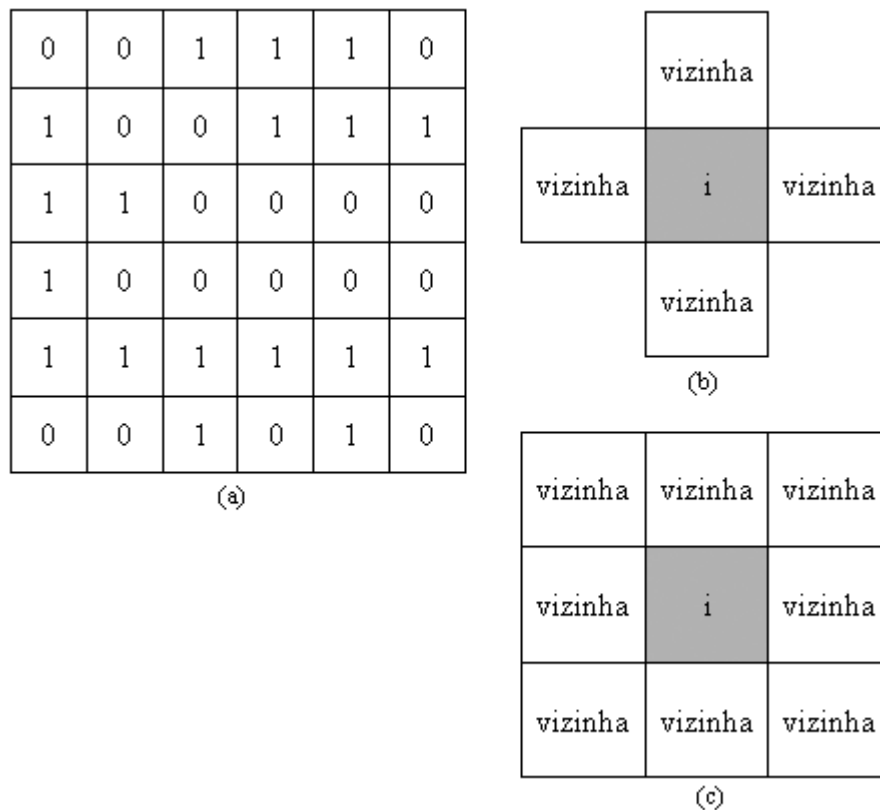


Fig. 3.5: (a) Exemplo de reticulado de um AC binário bidimensional, (b) a vizinhança de von Neumann e (c) a vizinhança de Moore.

Um dos ACs bidimensionais mais conhecidos é o *Game of Life* (também conhecido apenas por *Life*), proposto por John Conway no final dos anos 60 [9]. O *Life* é um AC bidimensional binário que adota o modelo de vizinhança proposto por Moore (Figura 3.5c). A regra de transição é definida da seguinte forma:

- Uma célula que está no estado 1 permanecerá neste estado se 2 ou 3 vizinhas estiverem no estado 1; caso contrário, ela irá para o estado 0.
- Uma célula que está no estado 0 irá para o estado 1 se exatamente 3 vizinhas estiverem no estado 1; caso contrário, ela permanecerá em 0.

O *Life* começa com uma configuração inicial com um número finito de 1s. Uma célula cujo estado é 0 é chamada de célula *morta* e uma célula de estado 1 é chamada de célula *viva*. Este AC é muito conhecido porque, com frequência, dá origem a padrões interessantes e complexos no espaço de células, apesar de sua regra de transição ser simples. Por exemplo, a Figura 3.6 apresenta um tipo de estrutura interessante encontrada no *Life*, a qual se assemelha a um relógio.

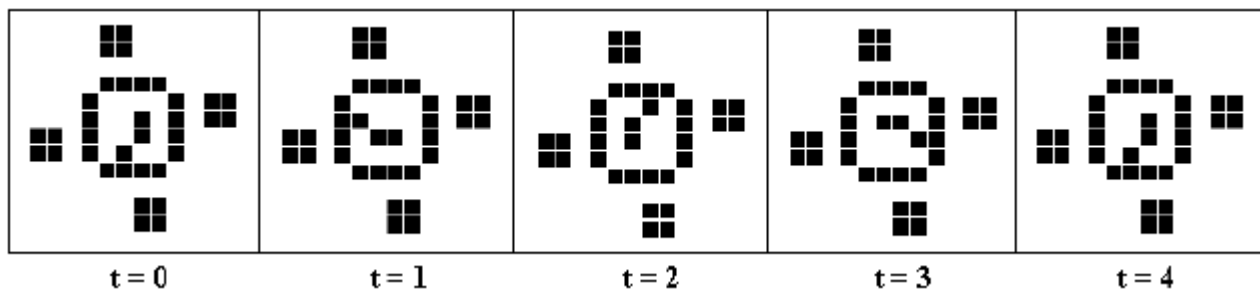


Fig. 3.6: Estrutura periódica no *Life* semelhante a um relógio [28]

O *Life* é um dos exemplos mais simples da chamada “complexidade emergente” ou “sistemas de auto-organização”. Esta área tem chamado a atenção de matemáticos e cientistas em diversos campos. Ela é o estudo de como elaborar padrões e comportamentos que possam emergir de regras muito simples. Ela nos ajuda a entender, por exemplo, como as pétalas de rosa ou as listras em uma zebra podem surgir de um tecido de células vivas crescendo juntas. Ela pode até mesmo nos ajudar a entender a diversidade da vida que se desenvolveu no planeta [17].

3.4 Dinâmica

Os ACs elementares, apresentados na Seção 3.3, são os tipos de AC mais simples conhecidos na literatura. Mesmo diante desta simplicidade, eles podem exibir diferentes tipos de comportamento dinâmico. Os ACs se tornaram importantes objetos de estudo para a área de sistemas dinâmicos. Uma característica pouco comum dos ACs é que eles são exemplos de sistemas dinâmicos totalmente discretos, uma vez que tanto os estados das células quanto o tempo e o espaço são discretos.

O método mais direto para se analisar a dinâmica de um AC é observar o comportamento dos diagramas de padrões espaço-temporais que ele gera, como o apresentado na Figura 3.4, partindo-se de várias configurações iniciais do reticulado. Após observado, este comportamento pode ser quantificado através da utilização de métricas estatísticas, como por exemplo, a informação mútua, a entropia e a taxa de espalhamento de padrões de diferença [66, 119].

Outro método utilizado é a análise da regra de transição do AC, que tem a vantagem de o tamanho de uma regra ser pequeno, comparado ao tamanho do reticulado. Diante deste fato, grande parte das pesquisas sobre a dinâmica dos ACs é voltada para o estudo do espaço formado por todas as regras de transição possíveis, fixadas a dimensão, o número de estados e o raio do AC [11, 12, 32, 66]. A regra de transição do AC determina, a partir do estado corrente do reticulado, qual será o próximo estado. Esta regra é a maior determinante da dinâmica observada na evolução de um AC, sendo tal dinâmica dependente também da configuração inicial do reticulado. O espaço de regras é formado pelo conjunto de todas as regras possíveis (k^n) com comprimento n . Por exemplo, sendo $k = 2$ e $r = 2$, o tamanho da regra n é 32 e o espaço de regras é formado por 2^{32} regras diferentes entre si. Existem diferentes esquemas de classificação do comportamento dinâmico de um AC. O mais conhecido foi proposto por Wolfram [121] e divide o AC em 4 classes:

Tipo 1 (Ponto Fixo): a configuração limite é invariante ao replicarmos a regra do AC (Figura 3.7a).

Tipo 2 (Periódicas): a configuração limite é invariante à aplicação da regra L vezes, sendo $L \geq 2$, com o tamanho do ciclo L sendo independente ou fracamente dependente do tamanho do sistema (Figura 3.7b, sendo $L = 2$).

Tipo 3 (Caóticas): produzem dinâmicas não periódicas. Estas regras são caracterizadas pela divergência exponencial do comprimento do seu ciclo com o tamanho do sistema e pela instabilidade

com respeito a perturbações (Figura 3.7c).

Tipo 4 (Complexas): embora a dinâmica limite possa ser periódica, o intervalo de transição pode ser longo e tipicamente este intervalo cresce mais que linearmente com o tamanho do sistema (Figura 3.7d). O AC complexo mais conhecido é o *Game of Life*, discutido na Seção 3.3.2.

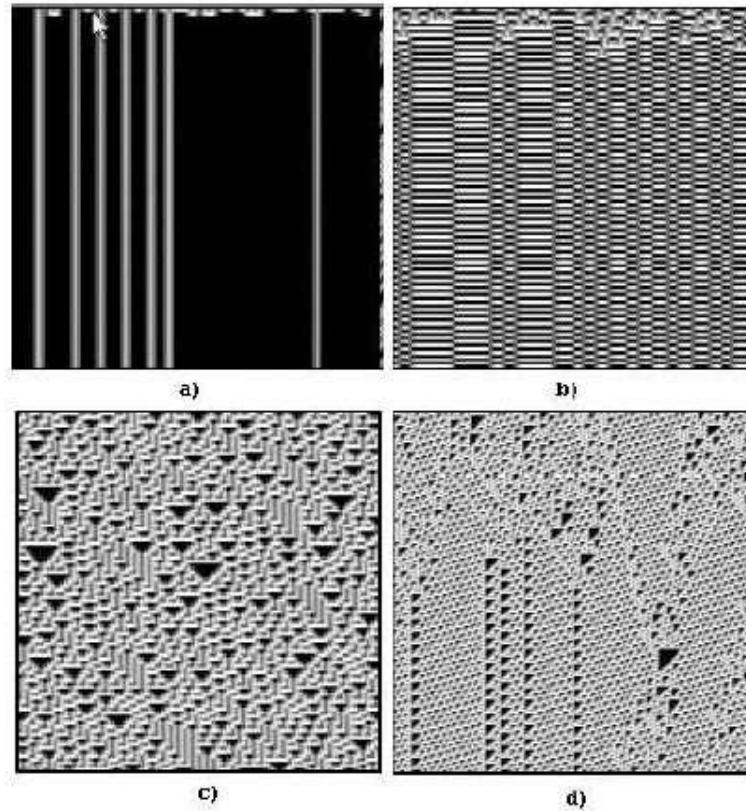


Fig. 3.7: Classificação do comportamento dinâmico dos ACs elementares: (a) Regra 36, (b) Regra 37, (c) Regra 30 e (d) Regra 110 [28].

Os pesquisadores têm se empenhado no estudo e na elaboração de parâmetros utilizados na previsão do comportamento dinâmico dos ACs. Esses parâmetros são calculados de forma estática diretamente a partir da regra de transição. Dentre os parâmetros citados na literatura, o parâmetro λ , proposto por Langton na década de 90, é o mais referenciado [27]. A seguir, são apresentados os principais conceitos dos parâmetros λ , Sensitividade e Domínio da Vizinhança [28].

Parâmetro λ : Langton propôs um parâmetro muito simples que foi chamado de λ [27], também conhecido como atividade [63]. Considerando-se os ACs binários, λ é simplesmente a fração

de 1s nos bits de saída da regra de transição.

Sensitividade: O parâmetro sensibilidade μ foi proposto por Binder [11, 12] e tem a finalidade de contar o percentual de configurações semelhantes que são mapeadas para estados diferentes. A sensibilidade é dada pelo número de mudanças na saída da função de transição causadas pela mudança no estado de apenas uma célula da vizinhança, tomado ao longo de todas as células dentre as possíveis vizinhanças da regra em questão.

Domínio da Vizinhança: A finalidade deste parâmetro é quantificar as mudanças provocadas pela regra de transição do AC, no estado da célula central, em relação ao estado que predomina na vizinhança como um todo. Ele verifica se o novo valor da célula central, determinado pela regra de transição, “acompanha” o estado que domina a configuração da vizinhança [33].

3.5 Computação em ACs

Assim como os ACs têm o potencial de modelar o comportamento de sistemas naturais, eles também têm a habilidade de executar computações complexas com um alto grau de eficiência e robustez [28]. No entanto, entender com os autômatos celulares são capazes de computar é algo ainda extramente vago e tem feito com que vários pesquisadores busquem métodos para viabilizar a sua programação.

Os estudos sobre o poder computacional dos ACs podem ser divididos em duas abordagens distintas: a computação explícita e a computação implícita [69]. Na computação explícita, busca-se encontrar modelos que associem a dinâmica dos ACs com a Máquina de Turing. Já na computação implícita, a computabilidade do AC é executada diretamente da regra do AC, onde a configuração inicial do reticulado é a entrada da computação e a saída é dada pela configuração do reticulado após um certo tempo. Ou seja, a computabilidade do AC é considerada como uma decorrência natural do seu comportamento dinâmico complexo.

O algoritmo de escalonamento baseado em ACs discutido nessa dissertação é um exemplo de computação implícita. Neste algoritmo proposto originalmente em [94], é assumido que cada tarefa de um programa paralelo [95] é associada a uma célula de um AC binário unidimensional. Nesse algoritmo, tanto o reticulado quanto a regra de transição são binários. O estado 0 associado a uma célula do reticulado significa que a tarefa correspondente está alocada no processador P0 e o estado

1 significa a alocação no processador P1. Assim, o reticulado do AC representa uma distribuição das tarefas entre os dois processadores. A regra de transição depende dos estados das células associadas com as tarefas vizinhas do programa paralelo. Diferentes modelos de vizinhança podem ser utilizados nesse algoritmo. No modelo mais simples, denominado Vizinhança Linear, a vizinhança é similar à descrita na Seção 3.3.1, onde o conceito de raio é utilizado. Nos outros dois modelos ditos não-lineares, a vizinhança não depende da posição da tarefa no reticulado, mas é baseada no conjunto de tarefas próximas da tarefa em questão, no programa paralelo que se deseja escalonar. O propósito do algoritmo de escalonamento é encontrar regras de transição de ACs que, para qualquer configuração inicial do reticulado, possam levar o mesmo a convergir para uma configuração (distribuição das tarefas entre os processadores P0 e P1) que minimize o tempo total de execução T do programa paralelo. A busca das regras é feita através de um algoritmo genético. O algoritmo de escalonamento será discutido em detalhes no Capítulo 4.

Outro exemplo de computação implícita em ACs é o uso de regras de transição como métodos criptográficos [29, 47]. A próxima seção descreve o exemplo mais conhecido de estudo da computação implícita dos ACs, aliada a utilização de técnicas evolutivas, para a solução da Tarefa de Classificação da Densidade [69, 71, 72].

3.6 Tarefa de Classificação da Densidade

Formalmente, a Tarefa de Classificação da Densidade (TCD) [69, 71, 72] é chamada Tarefa “ $\rho_c = \frac{1}{2}$ ”, onde ρ denota a densidade de 1s na configuração de um AC binário, ρ_c denota uma densidade crítica, ou limite para a classificação e ρ_0 denota uma densidade de 1s na configuração inicial do AC. Se $\rho_0 > \rho_c$, então em T passos de tempo o AC deve convergir para uma configuração ponto-fixa de 1s (ou seja, todas as células permanecerão no estado 1). Caso contrário, deve convergir para uma configuração ponto-fixa de 0s. O número de passos T é um parâmetro da TCD que depende do tamanho N do reticulado. Informalmente, o objetivo da TCD é encontrar um AC binário unidimensional que decida se uma configuração inicial de 0s e 1s possui uma quantidade maior de 1s. Se for maior, o reticulado deste AC deve convergir para uma configuração ponto-fixa de 1s; caso contrário, deve convergir para uma configuração ponto-fixa de 0s.

A Figura 3.8 apresenta duas evoluções de um AC executando a TCD. Neste AC, o raio r é igual a 3 e o reticulado é composto de $N = 149$ células. Na primeira evolução (esquerda), a configuração inicial do AC possui um quantidade de 1s menor que a quantidade de 0s ($\rho_0 < \frac{1}{2}$). Observa-se que, na configuração final (decorridos 300 passos de tempo), todas as células do reticulado estão no estado 0 (cor branca). Na segunda evolução, que possui uma quantidade de 1s maior que a de 0s ($\rho_0 > \frac{1}{2}$), observa-se que na configuração final do AC todas as células estão no estado 1 (cor preta).

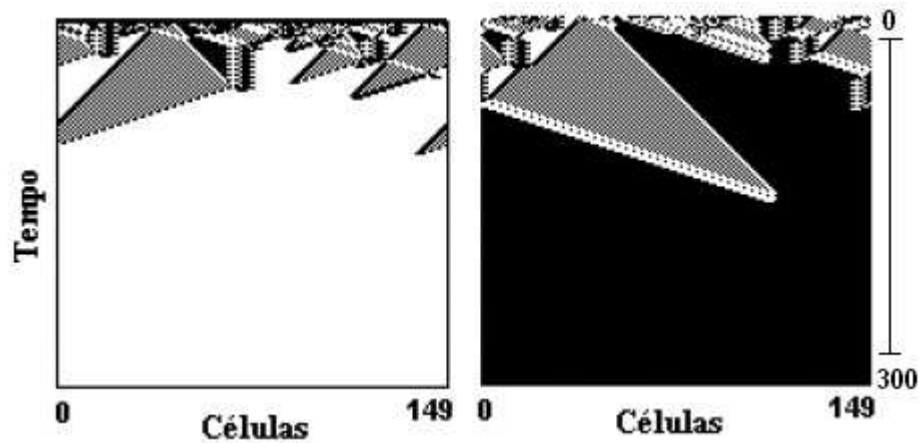


Fig. 3.8: Diagrama de padrões espaço-temporais para um AC que executou corretamente a TCD [28].

Algumas regras de transição apresentadas na literatura, capazes de resolver o problema da TCD, foram encontradas manualmente [27], enquanto em outras abordagens, regras foram encontradas com a utilização de técnicas evolutivas, como por exemplo, os algoritmos genéticos (AGs) [72] e a Programação Genética (PG) [58].

Uma regra chamada GKL, obtida manualmente e proposta por Gacs, Kurdyumov e Levin, foi apresentada, em 1978 [27]. O desempenho desta regra em 10^4 configurações iniciais diferentes foi de 81,6%. Em 1995, Mark Land e Richard Belew provaram que não existe regra de transição de ACs que consiga, com 100% de desempenho, realizar a TCD, para nenhum raio finito [61].

Mitchell, Hraber e Crutchfield apresentaram os resultados de seus experimentos, nos quais foram utilizados os algoritmos genéticos na busca de regras de transição de AC capazes que realizar a TCD [72]. O tamanho da população N_{pop} utilizado foi igual a 100 e o AG evoluiu por 100 gerações. A função de avaliação (aptidão) de cada cromossomo (indivíduo) media o desempenho do AC em classificar 100 configurações iniciais diferentes (CIs) do reticulado. Em aproximadamente 300 execuções,

a melhor regra encontrada teve um desempenho inferior à GKL (em torno de 76,9%). Posteriormente foi obtida uma melhoria significativa na evolução de AGs simples, através da incorporação de uma heurística baseada em parâmetros de previsão de comportamento dinâmico dos ACs (Seção 3.4). Nos experimentos realizados, com a utilização da heurística, o desempenho foi elevado de 76,9% para 80,4% [31].

A primeira regra de transição evoluída e que apresentou um resultado superior às regras manuais foi apresentada em [4]. A técnica evolutiva utilizada foi a Programação Genética (PG), implementada em uma arquitetura paralela de 64 nós, com um tamanho de população $N_{pop} = 51.200$ e 1.000 CIs diferentes utilizadas no cálculo da aptidão dos indivíduos. Em 5 execuções do programa, a melhor regra encontrada teve um desempenho de 82,3%.

Em [56], Juillé e Pollack divulgaram os resultados dos seus experimentos conduzidos com a utilização de um Algoritmo Genético Coevolutivo (Seção 2.10), utilizando a metáfora presa-predador e uma avaliação baseada na técnica de Recurso Compartilhado. Com esta abordagem, o algoritmo evoluía duas populações diferentes: uma população de regras de ACs e outra composta por configurações iniciais (CIs). Realizando 6 execuções do AG, com um tamanho de cada população $N_{pop} = 1.000$ indivíduos, a melhor regra encontrada teve o desempenho de 86%, sendo esta a regra de maior desempenho para a Tarefa de Classificação da Densidade publicada até hoje [5].

A evolução de regras de transição de ACs no algoritmo de escalonamento proposto por Seredynski em [94], que será discutido no Capítulo 5, foi fortemente baseada nos experimentos citados anteriormente para a TCD, especialmente no AG padrão implementado por Mitchell e colegas [72].

Capítulo 4

Escalonamento de Tarefas em Multiprocessadores

Este capítulo discute o escalonamento de tarefas em sistemas multiprocessadores. O escalonamento das tarefas de um programa paralelo em uma arquitetura multiprocessadora, mesmo limitada a dois processadores, é um problema de difícil solução e constitui-se em um grande desafio para pesquisadores da área de computação paralela. Este problema é conhecido por ser NP-completo em sua forma geral [39]. A maioria dos algoritmos de escalonamento encontram soluções sub-ótimas, e alguns deles são aptos a encontrar soluções ótimas (exatas) apenas para alguns casos especiais do problema, os quais podem não representar completamente sistemas do mundo real.

Diferentes heurísticas baseadas em plataformas matemáticas e metaheurísticas baseadas em mecanismos observados na natureza têm sido propostos para resolver o problema do escalonamento. Dentre as heurísticas mais conhecidas podemos citar *list scheduling*, *critical path*, e *clustering* [36, 60]. E, dentre as metaheurísticas conhecidas e aplicadas com sucesso podemos citar *Tabu Search* (TS) [42], *Simulated Annealing* (SA) [1], *Algoritmos Genéticos* (AG) [43], *Redes Neurais* [53, 75, 87] e *Branch and Bound* [54].

Apesar destas heurísticas e metaheurísticas conseguirem, na maioria dos casos, resolver o problema do escalonamento, elas apresentam um alto consumo de tempo e/ou memória representado pelo custo de execução do escalonador [104]. No caso das metaheurísticas, a principal causa relacionada a este alto consumo é a necessidade de cálculo de uma *função de custo* durante várias iterações

do algoritmo de escalonamento.

Os algoritmos de escalonamento existentes não extraem, conservam ou reutilizam qualquer conhecimento sobre o problema enquanto resolvem instâncias do problema de escalonamento. Caso este conhecimento fosse “armazenado” e utilizado posteriormente na resolução do próprio problema de escalonamento ou de outras instâncias diferentes, poder-se-ia obter um ganho durante a resolução destas instâncias. Enquanto a maioria dos algoritmos de escalonamento são seqüenciais, uma nova área de pesquisa estuda algoritmos de escalonamento paralelos e distribuídos [3, 92], baseados na aplicação de técnicas computacionais não convencionais [93]. No Capítulo 5 é apresentado um algoritmo de escalonamento baseado em autômatos celulares (ACs) e algoritmos genéticos (AGs). Com este algoritmo, espera-se que o conhecimento sobre o processo de escalonamento possa ser extraído e reutilizado enquanto resolvendo novas instâncias do problema de escalonamento. Resultados recentes mostraram que, combinados, os ACs e os AGs podem ser efetivamente usados para projetar algoritmos de escalonamento paralelos e distribuídos [95, 96, 97, 98, 99, 102, 104].

Nesse capítulo, faremos uma revisão sobre os conceitos relacionados ao escalonamento em multiprocessadores e discutiremos métodos empregados atualmente na sua solução. A Seção 4.1 apresenta os principais conceitos e notações utilizadas para a representação de uma arquitetura multiprocessadora e de um programa paralelo. Na Seção 4.2 são apresentados alguns métodos baseados em heurísticas e na Seção 4.3 são apresentados outros baseados em metaheurísticas, todos utilizados no escalonamento de tarefas em multiprocessadores.

4.1 Conceitos e Notações

Um sistema multiprocessador é representado por um grafo não-direcionado e acíclico $G_s = (V_s, E_s)$ chamado *grafo de sistema*. V_s é o conjunto de N_s nós do grafo de sistema que representam cada processador com as suas memórias locais de um computador paralelo de uma arquitetura MIMD¹. E_s é o conjunto de *arestas* representando canais bidirecionais entre os processadores e define uma topologia de um sistema multiprocessador. A Figura 4.1 apresenta um exemplo de um sistema multiprocessador composto por dois processadores: P0 e P1. É assumido que todos os processadores têm

¹Máquinas MIMD (Multiple Instruction Multiple Data) são arquiteturas caracterizadas pela execução simultânea de múltiplos fluxos de instruções.

o mesmo poder computacional e a comunicação entre eles, através do *link*, consome um tempo de processamento desprezível.



Fig. 4.1: Grafo de sistema representando uma arquitetura com dois processadores

Um programa paralelo, também conhecido como *grafo de programa* ou *grafo de precedência de tarefas*, é representado por um grafo direcionado, ponderado e acíclico $G_p = (V_p, E_p)$. V_p é o conjunto de N_p nós do grafo que representam as tarefas elementares, as quais são unidades computacionalmente indivisíveis. E_p é o conjunto de *arestas* do grafo de programa que descreve o padrão de precedência e o custo de comunicação entre as tarefas. Existe uma relação de restrição de precedências entre as tarefas k e l no grafo de programa se o resultado produzido pela tarefa k tem que ser enviado para a tarefa l . Isto significa que a tarefa l só poderá iniciar a sua execução, em um determinado processador, após o término da tarefa k .

Uma tarefa é uma unidade indivisível de computação, a qual pode ser uma indicação de atribuição, uma subrotina ou até mesmo um programa completo. Considera-se, nessa dissertação, que as tarefas são *não-preemptivas*, o que significa que uma vez que uma tarefa tenha começado sua execução, ela continua até o término de sua execução, sem ser interrompida [90].

Um *macro fluxo de dados estático* [35, 90] é um modelo de execução do grafo de programa, sendo similar ao modelo de fluxo de dados. Os dados fluem através do grafo de programa e uma tarefa espera receber todos dos dados em paralelo antes de iniciar sua execução. Assim que a tarefa completa sua execução ela envia, paralelamente, o dado de saída para todas as suas tarefas sucessoras.

Um grafo de programa possui dois atributos: pesos b_k e a_{kl} . Os pesos b_k dos nós (tarefas) representam o tempo necessário (custo computacional) para a execução de uma dada tarefa k em qualquer processador de um dado sistema multiprocessador. Os pesos a_{kl} das *arestas* descrevem o tempo de comunicação (custo de comunicação) entre pares de tarefas k e l , quando elas estão localizadas em processadores distintos. O custo de comunicação representa o tempo que a tarefa l deve aguardar para iniciar a sua execução, após o término da tarefa k .

A Figura 4.2 apresenta o grafo de programa conhecido como Gauss18 [96]. Ele é composto

por 18 tarefas e representa o algoritmo de eliminação Gaussiana ². O código interno de cada nó do grafo de programa representa o *número de ordem* da tarefa, isto é, a identificação da tarefa. Os números colocados ao lado esquerdo de cada nó representam o seu custo computacional, ou seja, quantas unidades de tempo são gastas na execução da tarefa em qualquer processador. Os números colocados nas arestas (setas) representam o custo de comunicação entre as tarefas k e l quando elas estão alocadas em processadores distintos. Considere, por exemplo, a tarefa 9. Esta tarefa tem um custo computacional igual a 3, ou seja, serão gastas 3 unidades de tempo para executar esta tarefa em qualquer processador. Caso 9 e 13 estejam alocadas em processadores distintos, após o término da tarefa 9 a tarefa 13 deverá aguardar, no mínimo, 8 unidades de tempo para iniciar a sua execução, correspondente ao atraso na transmissão dos dados obtidos na tarefa 9.

O *escalonamento* é definido como o processo de distribuição das tarefas de um grafo de programa entre os p processadores de um grafo de sistema, de forma de que as restrições de precedência sejam preservadas, e o *tempo total de execução* T seja minimizado. Geralmente, para representar uma solução encontrada para um dado grafo de programa, é utilizado um *Gráfico de Gantt* que mostra a alocação das tarefas nos processadores e o tempo em que começa e termina a execução de cada uma das tarefas. A Figura 4.3 mostra um gráfico de Gantt que representa uma alocação ótima do Gauss18 no grafo de sistema da Figura 4.1, composto pelos processadores P0 e P1.

O termo *clustering* significa o mapeamento das tarefas em *clusters*. Um *cluster* é um conjunto de tarefas que serão executadas no mesmo processador. *Clusters* não são tarefas, e tarefas que pertencem a um *cluster* podem se comunicar com tarefas de outros *clusters* imediatamente após o término de sua execução. O *clustering* é um problema idêntico ao escalonamento no caso de um número ilimitado de processadores completamente conectados. Assim como o escalonamento, o *clustering* também é considerado, em sua forma geral, um problema NP-completo [19, 90].

Um *clustering* é chamado *não-linear* se duas tarefas independentes são mapeadas no mesmo *cluster*, caso contrário, ele é chamado *linear*. A Figura 4.4b apresenta um *clustering* linear com três *clusters* $\{n_1, n_2, n_7\}$, $\{n_3, n_4, n_6\}$ e $\{n_5\}$. A Figura 4.4c apresenta um *clustering* não-linear com *clusters* $\{n_1, n_2\}$, $\{n_3, n_4, n_5, n_6\}$ e $\{n_7\}$. Por exemplo, na Figura 4.4c as tarefas n_4 e n_5 são

²Em matemática, a eliminação Gaussiana é um algoritmo em álgebra linear para a determinação de soluções de um sistema de equações lineares. Formalmente, o algoritmo de eliminação Gaussiana recebe um sistema arbitrário de equações lineares como entrada e retorna o seu vetor solução, se ele existe e é único [116].

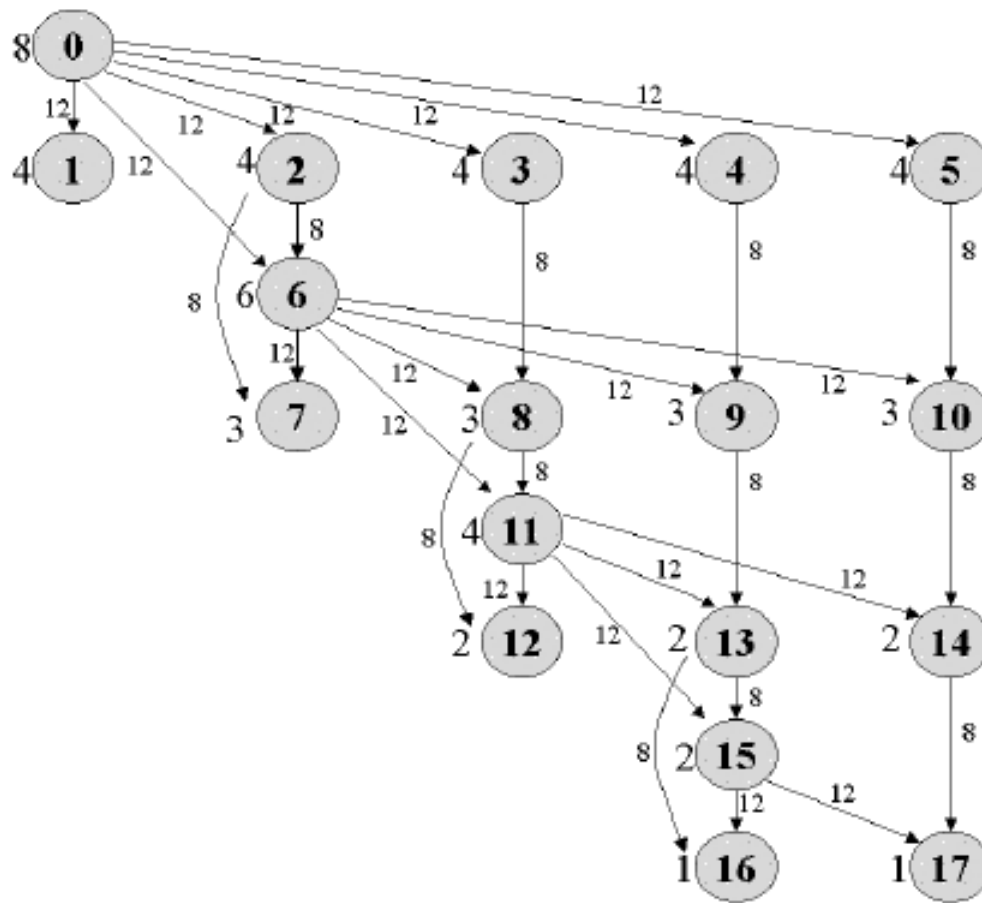
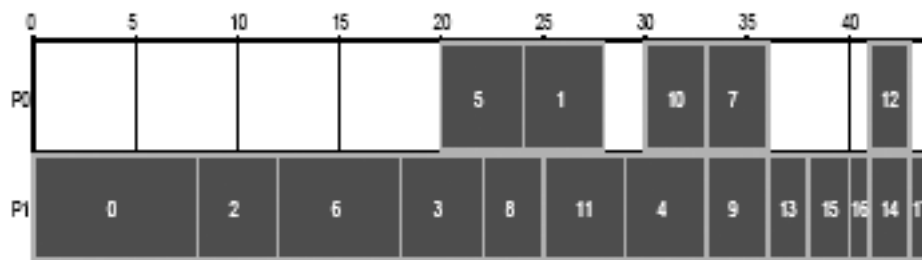


Fig. 4.2: Grafo de programa Gauss18

Fig. 4.3: Um exemplo de solução ótima para o Gauss18 ($T = 44,0$) [96].

independentes e estão mapeadas no mesmo *cluster*.

Para cada nó k do grafo de programa podem ser definidos conjuntos de tarefas *predecessoras*(k), *irmãs*(k) (isto é, tarefas que têm, no mínimo, uma predecessora em comum) e *sucessoras*(k). Por exemplo, para a tarefa 6 do Gauss18, apresentada na Figura 4.2, tem-se os conjuntos: *predecessoras*(6) = {0,2}, *irmãs*(6) = {1,2,3,4,5,7} e *sucessoras*(6) = {7,8,9,10,11}. Observe que a tarefa 7 é

considerada irmã e sucessora de 6, ao mesmo tempo, pois ela depende da saída de 6, mas também depende da saída de 2, que é predecessora de 6. Situações como esta caracterizam a estrutura irregular do grafo de programa Gauss18. As tarefas que não possuem predecessoras são chamadas *tarefas de entrada* e aquelas que não possuem sucessoras são chamadas *tarefas de saída*.

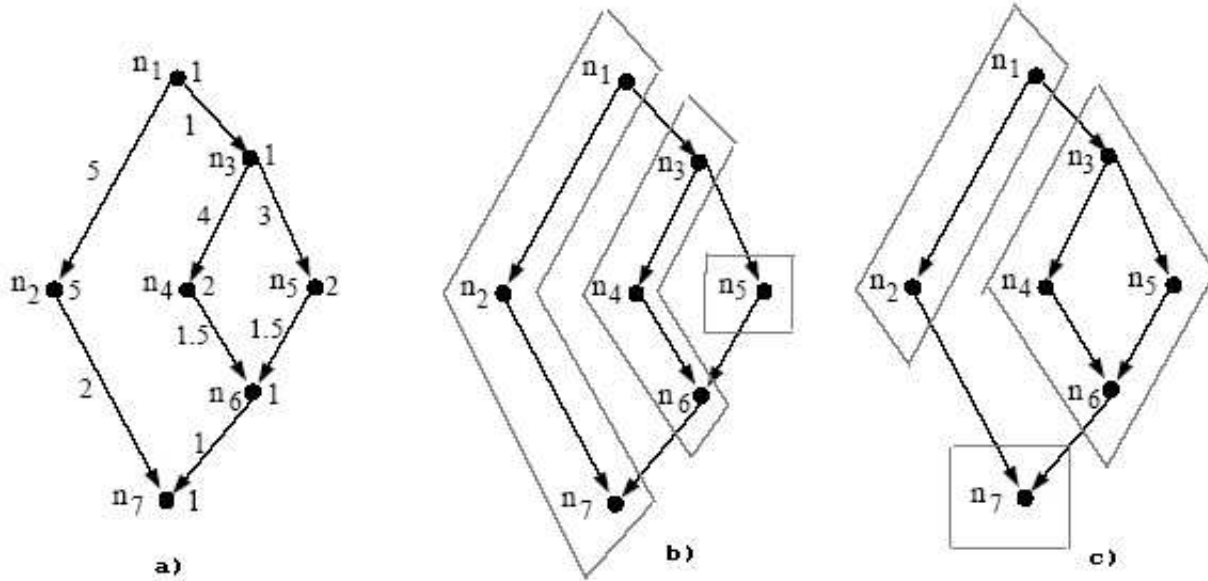


Fig. 4.4: Exemplo de *clustering*: (a) um grafo de programa, (b) um *clustering* linear e (c) um *clustering* não-linear [41].

Além do custo computacional b_k e do custo de comunicação a_{kl} , outros dois atributos adicionais podem ser definidos para cada tarefa de um grafo de programa: o *nível* e o *co-nível*. O nível h_k de uma tarefa k é o comprimento máximo do caminho mais longo de uma tarefa k para uma tarefa de saída, considerando os custos de comunicação de todas as arestas do caminho. Matematicamente, o nível h_k é definido como:

$$h_k = \begin{cases} b_k, & \text{para uma tarefa de saída} \\ \max_{l \in \text{sucessoras}(k)} (h_l + a_{kl}) + b_k, & \text{para outras tarefas} \end{cases} \quad (4.1)$$

Por exemplo, no Gauss18, a tarefa 1 tem $h_1 = 4$ (tarefa de saída), a tarefa 0 tem $h_0 = 102$ (caminho mais longo: $0 \rightarrow 2 \rightarrow 6 \rightarrow 8 \rightarrow 11 \rightarrow 13 \rightarrow 15 \rightarrow 16$) e a tarefa 6 tem $h_6 = 70$ (caminho mais longo: $6 \rightarrow 8 \rightarrow 11 \rightarrow 13 \rightarrow 15 \rightarrow 16$). O co-nível d_k é o comprimento do caminho mais longo de uma

tarefa de entrada para a tarefa k , considerando todos os custos de comunicação de todas as arestas do caminho. Matematicamente, o co-nível d_k é definido como:

$$d_k = \begin{cases} b_k, & \text{para uma tarefa de entrada} \\ \max_{l \in \text{predecessoras}(k)} (d_l + a_{lk}) + b_k, & \text{para outras tarefas} \end{cases} \quad (4.2)$$

Por exemplo, no Gauss18, a tarefa 0 tem $d_0 = 8$ (tarefa de entrada), a tarefa 1 tem $d_1 = 24$ (caminho mais longo: $0 \rightarrow 1$) e a tarefa 6 tem $d_6 = 38$ ($0 \rightarrow 2 \rightarrow 6$). O nível e o co-nível de uma dada tarefa k são valores estáticos e não dependem da alocação de um grafo de programa em um grafo de sistema. Entretanto, se estes valores forem calculados para tarefas alocadas nos processadores de um sistema paralelo, os valores do nível e do co-nível serão dependentes da alocação e serão chamados *nível dinâmico* e *co-nível dinâmico*, respectivamente. Nesse caso, tanto para o cálculo do nível dinâmico, quanto para o cálculo do co-nível dinâmico, os custos de comunicação nem sempre serão adicionados no cálculo do comprimento do caminho máximo. A adição dos mesmos dependerá se, no caminho, duas tarefas k e l , dependentes uma da outra, não estão no mesmo processador. Se elas estiverem alocadas no mesmo processador, o custo de comunicação é 0 (zero).

A função que calcula o tempo total de execução T para um dado grafo de programa tem como parâmetros uma *alocação* das tarefas em um grafo de sistema, isto é, uma distribuição das tarefas entre os processadores, e uma *política de escalonamento* [99], que define a ordem de execução das tarefas em cada processador, aplicada individualmente em cada processador. Ou seja,

$$T = f(\text{alocacao}, \text{politica_escalonamento}). \quad (4.3)$$

Seja, por exemplo, o grafo de sistema da Figura 4.5a, composto por dois processadores, e o grafo de programa da Figura 4.5b, composto por quatro tarefas. Suponhamos uma alocação em que as tarefas 0, 1 e 2 estejam alocadas no processador P0, e a tarefa 3 esteja no processador P1. Após a execução da tarefa 0, o processador P0 precisa decidir a próxima tarefa a ser executada. A Figura 4.5c apresenta o tempo total de execução T para o caso em que a próxima tarefa escolhida pela política de escalonamento foi a 1. Neste caso, obteve-se $T = 7$. A Figura 4.5d apresenta o valor de T para o caso em que a tarefa escolhida pela política de escalonamento foi a 2, obtendo-se $T = 10$. Logo, para uma

mesma alocação das tarefas e políticas de escalonamento diferentes, pode-se obter diferentes valores de T .

Para o algoritmo de escalonamento avaliado neste trabalho e que será descrito na Seção 5.1.4, assume-se que a política de escalonamento é invariante para uma dada execução do algoritmo, sendo a mesma para todos os processadores.

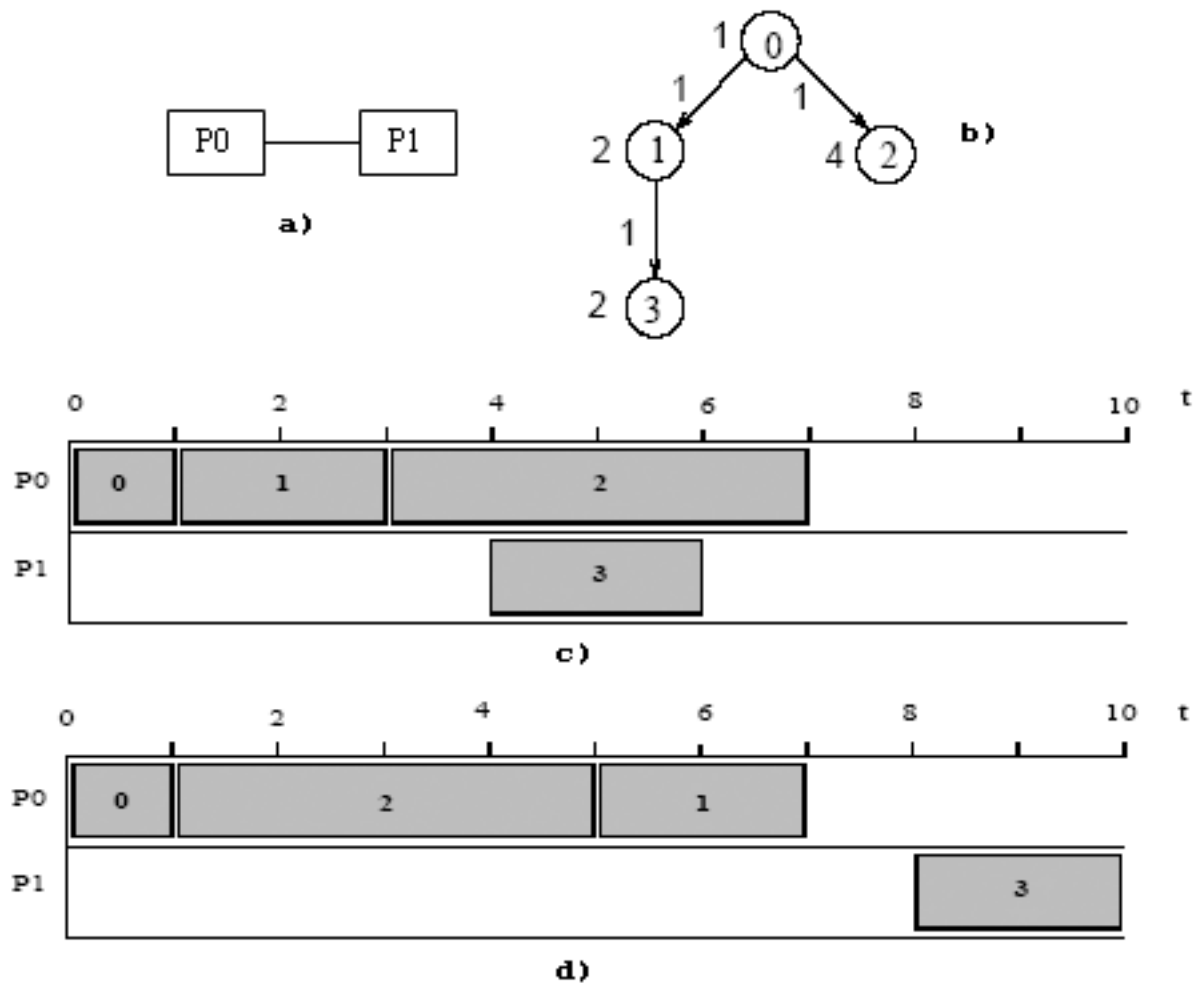


Fig. 4.5: Escalonamento multiprocessador: (a) Grafo de sistema, (b) grafo de programa, (c) e (d) escalonamento utilizando diferentes políticas de escalonamento, para a seguinte alocação nos processadores: tarefas 0, 1 e 2 em P0 e a tarefa 3 em P1.

4.2 Algoritmos de escalonamento baseados em heurísticas

Nesta seção, descrevemos alguns exemplos de algoritmos de escalonamento em multiprocessadores, os quais foram discutidos em [41]. Gerasoulis e Yang distinguiram duas classes de algoritmos de escalonamento [41]. Os métodos de um passo escalonam o grafo de programa diretamente em p processadores. Nestes métodos, as heurísticas consideradas têm as seguintes propriedades: 1) elas não duplicam as mesmas tarefas em dois processadores. 2) elas não recuam, ou seja, uma vez alocada uma tarefa em um processador, ela não poderá ser realocada mais a frente. Os métodos multi-passos primeiramente executam um passo de *clustering* no grafo de programa, supondo-se que há um número ilimitado de processadores completamente conectados, e então nos próximos passos os *clusters* são “fundidos” e escalonados em p processadores disponíveis.

4.2.1 Métodos de escalonamento de um passo

Em [41] são apresentados dois métodos de escalonamento de um passo. Basicamente, os dois métodos são variações do método de *escalonamento de lista*, sendo que o primeiro utiliza a heurística *Critical Path* (CP) e o segundo utiliza a heurística *Modified Critical Path* (MCP).

Escalonamento de lista com a heurística *Critical Path* (CP)

O método do escalonamento de lista escalona tarefas livres³ através da varredura, da esquerda para a direita, de uma lista de prioridades. Especificamente, os seguintes passos são seguidos:

1. Determine uma lista de prioridades.
2. Quando um processador está *disponível* para execução, percorra a lista da esquerda para a direita e escalone a primeira tarefa livre. Se dois processadores estão disponíveis, ao mesmo tempo, escalone a tarefa no processador com o menor número do processador.

Quando o custo de comunicação é igual a 0 (zero), uma boa heurística é a lista de prioridades do *Critical Path* (CP) [41]. A prioridade de uma tarefa é o seu *nível*, isto é, o comprimento do caminho

³Uma tarefa é dita *livre* se todas as suas predecessoras já foram executadas. Uma tarefa está *pronta* se ela é livre e todos os dados necessários para sua execução estão localmente disponíveis no processador onde a tarefa será escalonada.

mais longo do nó para um nó de saída. O escalonamento de lista com *Critical Path* processa muito bem quando o custo de comunicação é igual a 0. Por exemplo, ele é ótimo para grafos de programa em formato de árvore com custos iguais e para qualquer grafo de programa arbitrário com custos iguais escalonados em dois processadores [41]. Para qualquer outro tipo de grafo de programa ou um escalonamento em p processadores ($p > 2$), o escalonamento de lista com *Critical Path* não apresenta bons resultados. Quando o custo de comunicação é diferente de 0 (zero), a identificação de uma boa lista de prioridades é algo extremamente difícil [41]. Isto ocorre devido ao custo de comunicação tornar-se zero quando as tarefas ligadas pelo arco (aresta) de comunicação são escalonadas no mesmo processador, fazendo com que o cálculo do nível de prioridade seja dependente da alocação.

A Figura 4.6 exemplifica o comportamento da heurística CP no caso em que o nível das tarefas inclui tanto o custo computacional quanto o custo de comunicação. O grafo de programa da Figura 4.6a, será submetido ao escalonamento de lista, baseado na lista de prioridades com o maior nível primeiro. O nível da tarefa n_6 é 2 e o nível da tarefa n_3 é 4, o qual é igual ao nível máximo de todas as tarefas sucessoras, que é o somatório do nível de n_6 (igual a 2), do custo de comunicação entre n_3 e n_6 (igual a 1) e o custo computacional de n_3 (igual a 1). A maneira de se calcular o nível para cada tarefa do grafo de programa é dada pela Equação 4.1. Calculando-se os níveis para todas as tarefas, a lista de prioridades resultante é $\{n_1, n_2, n_5, n_4, n_3, n_6, n_7\}$. A Figura 4.6b apresenta o escalonamento conforme a heurística CP. No instante de tempo 0, as tarefas n_1 , n_2 e n_5 estão livres e os processadores P0 e P1 estão disponíveis. De acordo com a lista de prioridades, a tarefa n_1 é escalonada em P0 e no próximo passo a tarefa n_2 é escalonada no único processador disponível, P1. No instante 1, as tarefas n_3 , n_4 e n_5 estão livres e como a prioridade da tarefa n_5 é maior que as outras, ela é escalonada no processador P0 enquanto a próxima tarefa de maior nível, n_4 , é escalonada em P1. Estando n_4 e n_1 (sua predecessora) alocadas em processadores distintos, n_4 precisa esperar 4 unidades de tempo para então ficar pronta para iniciar sua execução no instante de tempo 5. A tarefa n_5 pode iniciar sua execução imediatamente visto que os dados que ela precisa estão naquele processador. As demais tarefas são escalonadas de forma similar, sendo o resultado final apresentado no Gráfico de Gantt da Figura 4.6b com $T = 10$.

Um problema com a heurística *Critical Path* quando o custo de comunicação é diferente de zero é que ela escala uma tarefa *livre* quando um processador se torna disponível, mesmo que esta tarefa

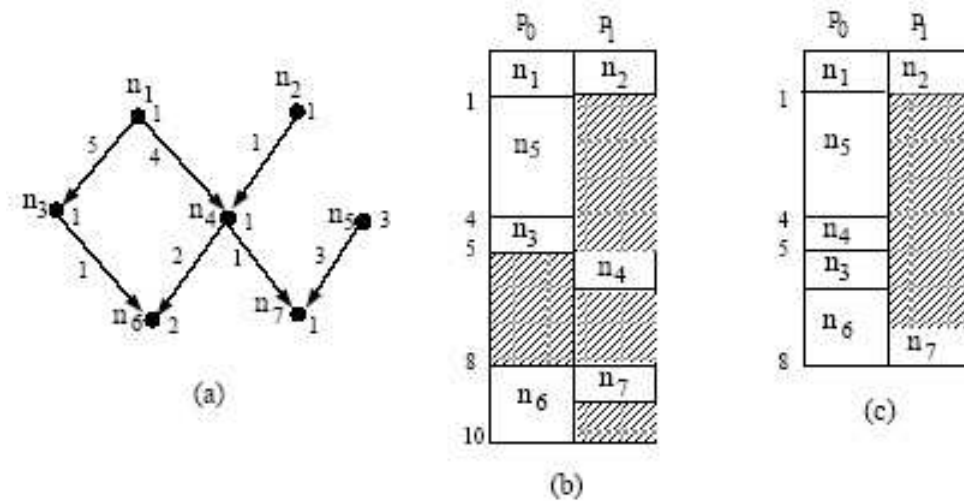


Fig. 4.6: Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP [41].

não esteja ainda *pronta* pra iniciar sua execução. Isto pode resultar em perda de performance do algoritmo, como mostra a Figura 4.7b. A tarefa n_3 é escalonada em P_1 uma vez que ela se torna livre no instante de tempo w . Entretanto, uma melhor solução é escalonar n_3 em P_0 como mostra a Figura 4.7c. A seguir é apresentada uma modificação da heurística CP, a heurística MCP.

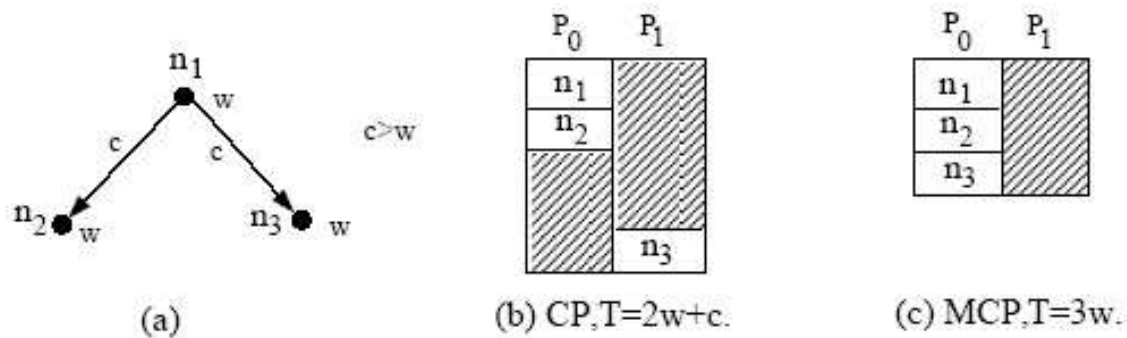


Fig. 4.7: Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP [41].

Escalonamento de lista com a heurística *Modified Critical Path* (MCP)

Diante dos problemas apresentados pelo método *Critical Path*, Wu e Gajski [126] apresentaram uma modificação neste algoritmo, propondo assim a heurística *Modified Critical Path*. Nela, ao invés de

escalonar uma tarefa livre para um processador disponível, a tarefa livre é escalonada no processador livre que permita que a tarefa inicie sua execução o mais cedo possível. A computação das prioridades usa novamente o maior nível, incluindo custos computacionais e de comunicação. Para o exemplo da Figura 4.7a, a lista de prioridades é $\{n_1, n_2, n_3\}$. O resultado do escalonamento desse grafo, a partir da heurística *Critical Path*, é apresentado na Figura 4.7b. O escalonamento a partir da heurística *Modified Critical Path* é mostrado na Figura 4.7c. A tarefa n_3 torna-se livre no instante de tempo w e ela é escalonada no processador P0 porque ela pode iniciar sua execução no tempo $2w$ que é mais cedo que o tempo $w + c$, desde que $c > w$.

Para o exemplo da Figura 4.6a, a lista de prioridades é a mesma que a apresentada para a CP: $\{n_1, n_2, n_5, n_4, n_3, n_6, n_7\}$. Depois que n_1, n_2 e n_5 são escalonadas, a tarefa n_4 tem a maior prioridade e está livre no tempo 2 mas ela não está pronta naquele instante a menos que ela seja escalonada em P0. Então, n_4 é escalonada em P0 no instante de tempo 4, ao invés de ser escalonada em P1, no instante de tempo 5. Neste caso, o tempo reduz para $T = 8$ como mostra a Figura 4.6c.

Mesmo que a heurística *Modified Critical Path* execute melhor que a heurística *Critical Path*, ela pode ainda apresentar perdas de performance em algumas situações [41]. Observe o exemplo do grafo de programa da Figura 4.8. O resultado do escalonamento com a heurística MCP é o mesmo do realizado com a heurística CP. Entretanto, se o custo de comunicação c é maior que o custo computacional w , o escalonamento ótimo para esse grafo é dado pela Figura 4.8d, no qual todas as tarefas são executadas no mesmo processador. A heurística MCP não pode reconhecer isto uma vez que ela usa o princípio do tempo de início mais cedo e ele inicia as tarefas n_2 e n_3 no tempo 0. Uma deficiência dos escalonamentos de um passo é que a prioridade das tarefas é dependente da alocação, porque o custo de comunicação entre tarefas será zero se elas estiverem alocadas no mesmo processador.

4.2.2 Métodos de escalonamento multipassos

Diante das limitações observadas nos métodos de escalonamento de um passo, diferentes abordagens foram propostas na literatura [57, 90]. Esses métodos são superiores no caso em que o escalonamento é realizado em grafos nos quais o custo de comunicação é diferente de zero. Nestas abordagens, o escalonamento é feito em mais de um passo. A seguir, apresentamos um desses métodos.

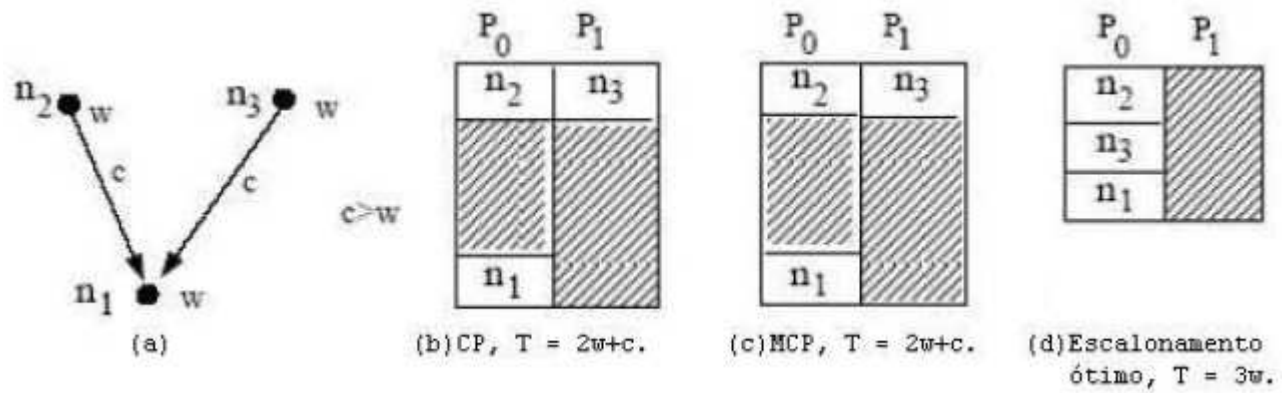


Fig. 4.8: Exemplo de CP e MCP: (a) grafo de programa, (b) o escalonamento através do CP, (c) o escalonamento através do MCP e (d) o escalonamento ótimo, se $c > w$ [41].

Em [40, 127], Gerasoulis e Yang propuseram um novo algoritmo de escalonamento chamado *Dominant Sequence Clustering (DSC)*. Esse algoritmo é baseado no conceito de *clustering*, apresentado na Seção 4.1, e na identificação e redução da *Dominant Sequence*, associada ao *clustering*. Uma *Dominant Sequence* é o caminho mais longo de um grafo de programa escalonado e dividido em *clusters* [41]. Este algoritmo é baseado na seguinte heurística:

- O tempo de execução T de um grafo de programa em um sistema com p processadores é determinado pela *Dominant Sequence (DS)*. Então, para reduzi-lo, deve-se zerar, no mínimo, um arco (aresta) na DS.
- Um algoritmo para redução da DS pode eliminar um ou mais arcos na DS ao mesmo tempo. Isto pode ser feito incrementalmente, em uma seqüência de passos.
- Uma eliminação de um arco será válida se o tempo de execução total T não é aumentado de um passo para o próximo.

O algoritmo DSC é uma variação do algoritmo DS, que executa todos os passos em uma complexidade de tempo “quase” linear no tamanho do grafo de programa [41]. A Figura 4.9 apresenta o refinamento feito com o algoritmo DSC. A Figura 4.9a mostra o *clustering* inicial do grafo de programa, no qual cada tarefa é associada a um *cluster* diferente. Na Figura 4.9a há duas *dominant sequences* com o tempo de execução $T = 10$: $\langle n_1, n_3, n_6 \rangle$ e $\langle n_1, n_4, n_6 \rangle$. No primeiro passo, o arco (n_1, n_3) da primeira DS é eliminado, colocando-se as duas tarefas em um único *cluster*, como

mostra a Figura 4.9b. Nesse passo, ainda resta a segunda DS $\langle n_1, n_4, n_6 \rangle$ com $T = 10$. Este passo é validado uma vez que o valor de T não é aumentado. No segundo passo, (n_1, n_4) é eliminado, incorporando-se a tarefa n_4 no *cluster* que já contém n_1 e n_3 . Após esse passo, o resultado são duas novas DS $\langle n_1, n_3, n_4, n_6 \rangle$ e $\langle n_5, n_7 \rangle$, como mostra a Figura 4.9c, com $T = 7$. Assim, o segundo passo é validado. No terceiro passo o arco (n_4, n_6) é eliminado, como mostra a Figura 4.9d, sendo esta eliminação também válida e com $T = 7$, determinado pela DS $\langle n_5, n_7 \rangle$. O próximo arco zerado é (n_5, n_7) , reduzindo o valor de T para 5. Finalmente, (n_2, n_4) e (n_4, n_7) não podem ser zerados porque o zeramento deles aumentará o valor de T . Então, os três *clusters* mostrados na Figura 4.9e são produzidos.

No terceiro passo mostrado na Figura 4.9c um algoritmo de ordenação é necessário para ordenar as tarefas no *cluster* não linear e então o tempo T deve ser computado para se obter a nova DS. Uma das principais idéias do algoritmo DSC é que ele computa o escalonamento e o tempo T incrementalmente.

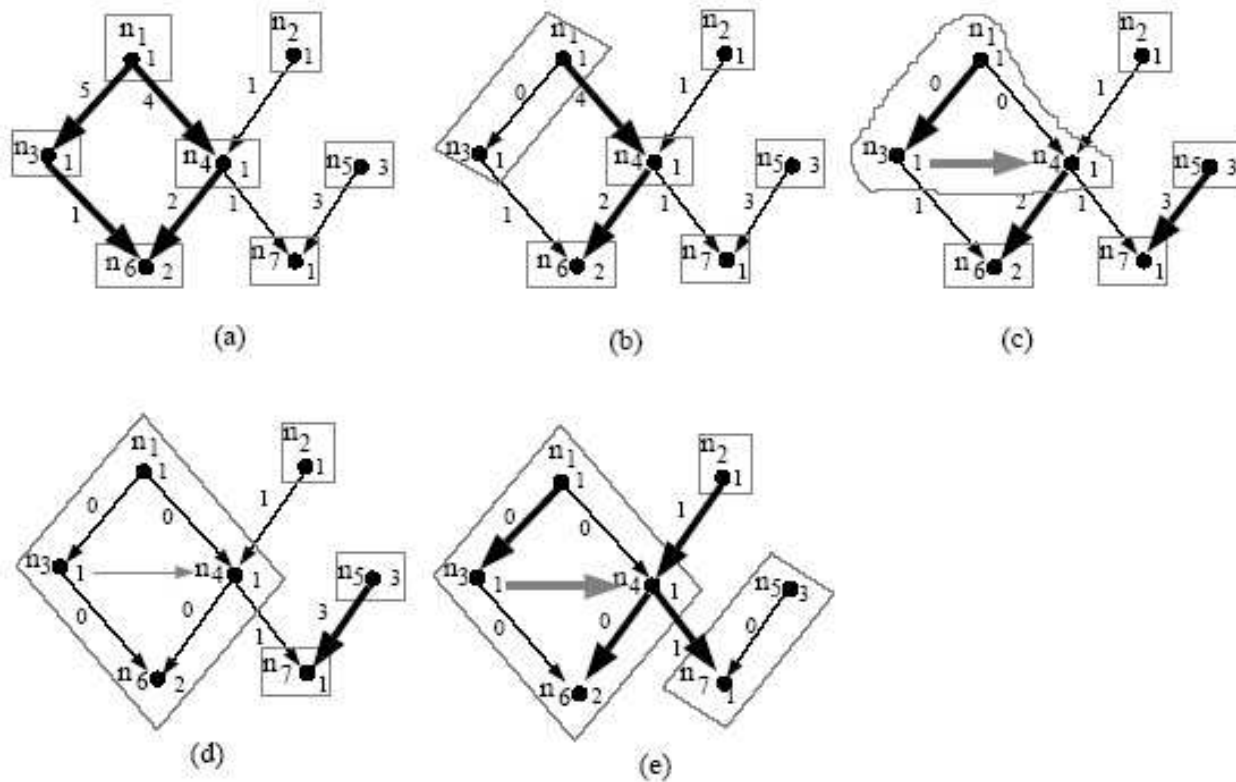


Fig. 4.9: Exemplo de um algoritmo DSC [41].

4.3 Metaheurísticas aplicadas ao problema do escalonamento de tarefas

Nesta seção, serão apresentados dois exemplos de algoritmos utilizados na resolução do problema do escalonamento de tarefas em sistemas multiprocessadores que são baseados em duas metaheurísticas: os algoritmos genéticos e a busca *branch and bound*.

4.3.1 Algoritmos genéticos aplicados ao escalonamento de tarefas

Por serem métodos probabilísticos e não exigirem muito conhecimento sobre os problemas para os quais são aplicados, os algoritmos genéticos são freqüentemente utilizados na tentativa de solucionar problemas complexos de otimização e aprendizado computacional. Em [89] foi realizado um estudo comparativo de algoritmos genéticos aplicados ao problema de escalonamento, utilizando-se duas abordagens distintas, onde foram testados diferentes maneiras de inicialização da população, função de avaliação, método de seleção, crossover (cruzamento), mutação e condição de término do AG. Nesta seção, como exemplo de aplicação de AGs para o problema do escalonamento de tarefas em multiprocessadores, descrevemos uma dessas abordagens.

O grafo de programa utilizado nos experimentos foi o Gauss18 (Figura 4.2) e uma arquitetura composta por dois processadores. Para a representação do problema (indivíduo) foi utilizada uma matriz dupla (2x18), onde são listadas cada tarefa e o processador (0 ou 1) nos quais elas estão alocadas. A Figura 4.12 ilustra dois exemplos de indivíduos.

Para a inicialização da população do AG, os indivíduos foram gerados aleatoriamente, sempre garantindo que cada tarefa aparecerá uma única vez no indivíduo, respeitando-se as restrições de precedência impostas pelo grafo de programa. Para cada tarefa também foi realizado o sorteio aleatório do processador ao qual ela estaria alocado, no caso, o processador 0 ou 1. Para se validar as restrições de precedência, a seguinte estratégia foi utilizada: para cada tarefa sorteada, verifica-se se as tarefas precedentes à sua execução já fazem parte do indivíduo. Caso esta condição tenha sido satisfeita, a tarefa passa a compor o indivíduo e é retirada da “lista” de tarefas a serem sorteadas. Caso contrário, a tarefa é descartada para este sorteio, mas continua sendo uma tarefa possível de ser selecionada. O sorteio de tarefas é feito até que o indivíduo esteja completo, ou seja, todas as tarefas alocadas.

A aptidão (*fitness*) do indivíduo é o tempo de execução das tarefas, seguindo a distribuição das tarefas entre os processadores e a ordem de execução em cada processador, ambas representadas no indivíduo. Por exemplo, são gastas 4 unidades de tempo na execução da tarefa 4. Caso a tarefa 4 e a tarefa 9 estejam alocadas em processadores diferentes, após o término da tarefa 4, a tarefa 9 deverá aguardar 8 unidades de tempo para iniciar sua execução. Caso eles estejam alocadas no mesmo processador, a tarefa 9 poderá, após o término da tarefa 4, imediatamente ser executada.

O método de seleção conhecido como *Torneio Simples* foi utilizado. Além da simplicidade de implementação, a escolha deste método foi feita pelo fato dele apresentar uma baixa pressão seletiva. Métodos com uma maior pressão seletiva poderiam causar uma convergência prematura da população, uma vez que a formação da população inicial utiliza um critério extremamente direcionado na geração do indivíduo. Neste torneio, três indivíduos da população são sorteados aleatoriamente, e o de menor *fitness* (menor tempo de execução das tarefas) é selecionado para participar de um par de pais do *crossover*. O indivíduo vencedor de um torneio é retirado e não participa da seleção do segundo pai.

Para realizar o cruzamento (*crossover*) entre os indivíduos pais selecionados, o método utilizado foi o *Crossover Cíclico*, que previne a existência de tarefas duplicadas no indivíduo, conforme apresentado na Figura 4.10. No indivíduo *Pai 1*, sorteia-se uma posição, e a tarefa nesta posição é passada para a mesma posição no novo indivíduo, o *Filho 2*. Para garantir que não haja repetição de tarefas nestes novos indivíduos, identifica-se no *Pai 2* a tarefa que foi selecionada no *Pai 1* e transfere-se esta para o *Filho 1*, na mesma posição em que foi encontrada, formando-se assim um ciclo, onde todos os genes dentro deste ciclo são transferidos da mesma maneira. Os genes que permanecem fora deste ciclo são copiados do *Pai 1* e *Pai 2* para as mesmas posições nos filhos *Filho 1* e *Filho 2*, respectivamente. Esse processo é aplicado considerando-se apenas as tarefas que compõem os indivíduos. Os processadores, de forma geral, acompanham as tarefas às quais estavam associados. Eventualmente, pode ocorrer uma mutação (alteração do processador), provocando uma maior “perturbação” no indivíduo.

Após realizado o cruzamento, o operador genético utilizado é a *Mutação*. O processo utilizado foi a *Permutação Simples*, que consiste no sorteio aleatório de duas posições (dois genes) no indivíduo, havendo a troca do conteúdo entre estas duas posições. A permutação simples é ilustrada na Figura 4.11. Essa mutação pode provocar indivíduos que não respeitem as precedências.

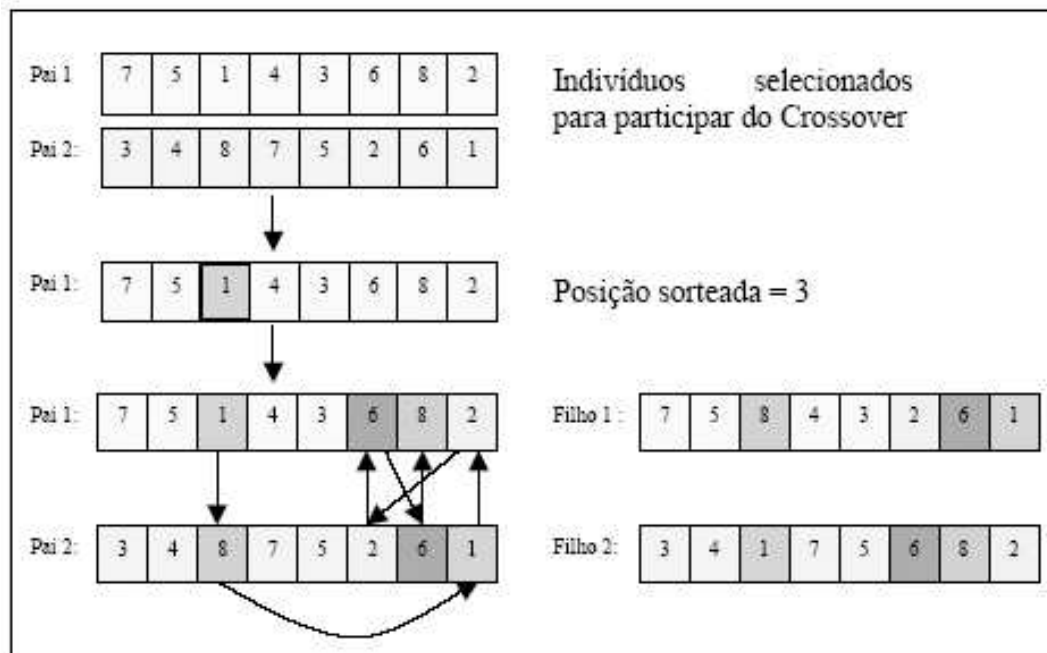


Fig. 4.10: Exemplo de crossover cíclico [89].

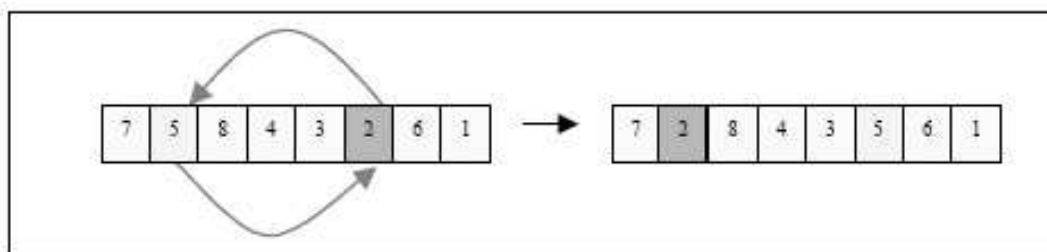


Fig. 4.11: Exemplo de permutação simples [89].

Foi utilizado como critério de término, o número máximo de 300 gerações. A taxa de indivíduos selecionados para o processo de *crossover* foi igual a 60%. A população foi formada por 250 indivíduos e a taxa de mutação por indivíduo foi de 40%.

O tempo total de execução T mínimo para o Gauss18 utilizando-se dois processadores é igual a 44 unidades de tempo. Existem diferentes soluções para este problema com $T = 44,0$. A abordagem descrita convergiu 7 vezes para uma solução ótima, em 10 execuções do AG. A Figura 4.12 apresenta dois exemplos de indivíduos obtidos, que retornam $T = 44,0$.

Tarefa	0	2	4	6	3	5	10	1	8	11	9	7	13	15	16	14	12	17
Processador	1	1	0	1	1	0	0	0	1	1	1	0	1	1	1	1	0	1

Tarefa	0	2	5	6	10	7	3	8	11	4	9	1	12	13	15	16	14	17
Processador	1	1	0	1	0	0	1	1	1	1	1	0	0	1	1	1	1	1

Fig. 4.12: Soluções ótimas obtidas com o AG no escalonamento do Gauss18 [89].

4.3.2 Algoritmo *Branch-and-Bound* aplicado ao escalonamento de tarefas em multiprocessadores

Os algoritmos *branch-and-bound* (B&B) são muito usados para resolver problemas combinatoriais. Estes algoritmos são baseados na idéia de enumerar todas as soluções *praticáveis* de um problema de forma direcionada e inteligente [16]. Uma solução é praticável se ela é uma solução completa “aceitável” para um determinado problema [55].

O B&B tem sido utilizado para encontrar soluções ótimas (ou sub-ótimas) para o problema do escalonamento de tarefas em sistemas multiprocessadores [82]. Ele é um eficiente algoritmo de busca no espaço de soluções para o problema de escalonamento. Este espaço de soluções é frequentemente representado por uma estrutura de árvore, onde cada vértice da árvore representa uma solução completa ou parcial para o problema. Utilizando-se de regras inteligentes para a seleção de vértices a explorar/expandir (*branch*) e também para a exclusão de vértices (*bound*) que não conduzam a uma solução ótima, a complexidade da busca pode ser reduzida quando comparada a métodos de busca enumerativa exaustiva. Ou seja, a escolha das regras de *branch* e *bound* são fundamentais para o sucesso do algoritmo.

Diante do fato da complexidade exponencial do algoritmo B&B não poder ser eliminada, a sua aplicabilidade, em geral, restringe-se a pequenos sistemas. Mas, quando as características da aplicação e/ou a arquitetura de processamento são conhecidas e podem ser exploradas eficientemente, o B&B pode ser bem aplicado mesmo para grandes sistemas [2, 52].

No método B&B, a busca por uma solução para o escalonamento multiprocessador é feita em uma *árvore de busca* que representa o espaço de soluções do problema, que são todas as possíveis permutações de atribuição tarefa-para-processador e ordem de execução das tarefas, isto é, a ordem de escalonamento. Cada vértice na árvore representa uma atribuição tarefa-para-processador, uma ordem

de execução das tarefas nos processadores. O *nó raiz* da árvore representa um escalonamento vazio e cada um de seus descendentes (vértices filhos) representa o escalonamento de uma tarefa específica em um processador específico. O filho de cada um destes vértices filhos representam o escalonamento de mais outra tarefa em um processador. A Figura 4.13a apresenta um grafo de precedência de tarefas composto por 4 tarefas e a Figura 4.13b apresenta um exemplo de árvore de busca para este grafo.

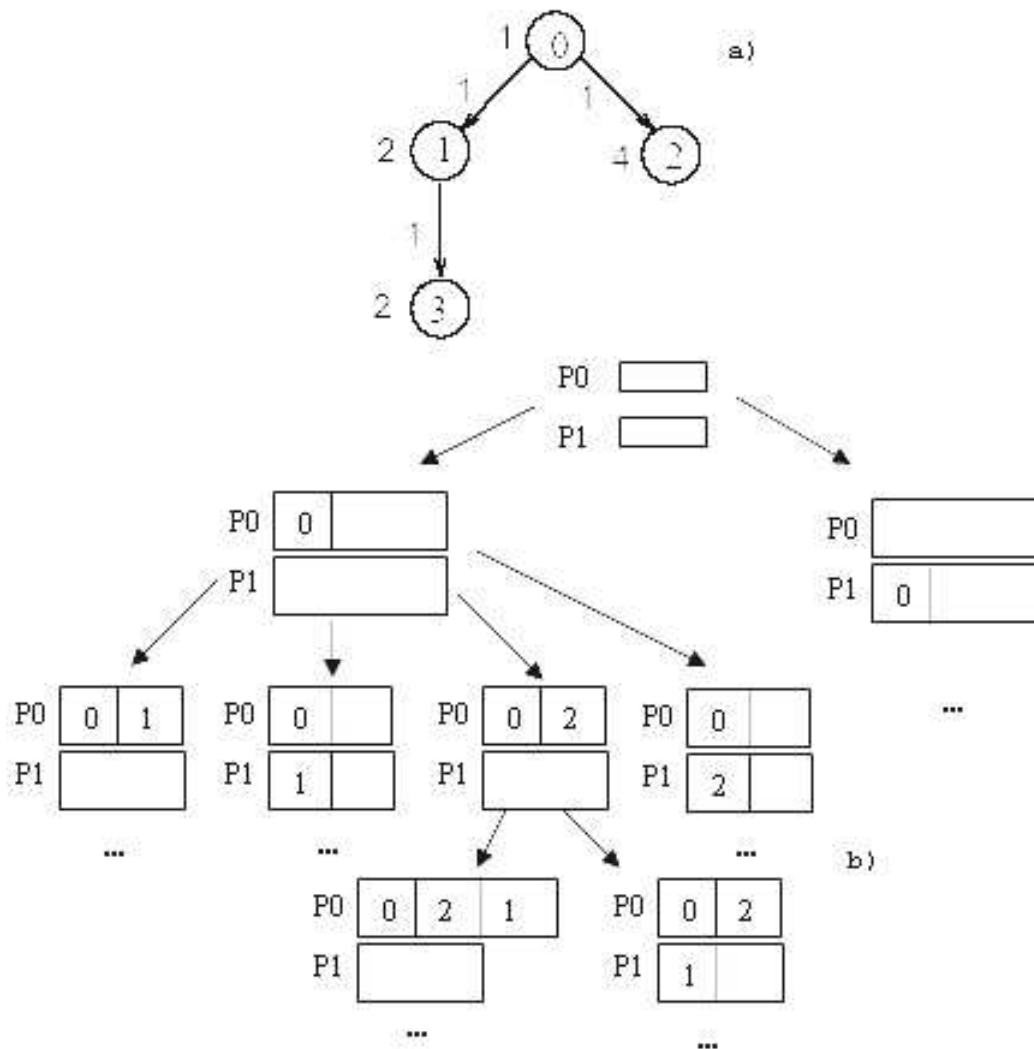


Fig. 4.13: Exemplo de árvore de busca: (a) grafo de precedência de tarefas e (b) árvore de busca.

Um *vértice objetivo* na árvore de busca representa uma solução completa onde todas as tarefas foram alocadas nos processadores. O *nível* de um vértice é o número de tarefas que foram atribuídas para qualquer processador no escalonamento (nó da árvore) corrente. O *custo* de um vértice é a

qualidade do escalonamento representado pelo vértice. Em [55], o custo é o atraso máximo de tarefa para o escalonamento representado pelo vértice.

Quando não há restrições de precedência entre as tarefas, considerando n tarefas e m processadores, o número de vértices objetivo na árvore de busca é $n!m^n$. Se as restrições de precedência são consideradas, o número de vértices objetivo na árvore de busca é reduzido para quase $k^n m^n$, sendo k o número máximo de vértices filhos de um vértice. Devido ao crescimento exponencial do número de vértices na árvore de busca, vértices não são, normalmente, gerados até que o algoritmo B&B necessite explorá-los. Quando um novo vértice é gerado e ele pode levar a uma solução ótima, ele é chamado de *vértice ativo*. Geralmente, uma heurística é utilizada para definir a ordem em que os vértices serão explorados na árvore de busca. A estratégia B&B consiste em alternar operações de *branching* e *bounding* no conjunto de vértice ativos. *Branching* (ramificando) é o processo de gerar vértices filhos de um vértice ativo, e *bounding* (avaliando) é o processo de avaliar o custo de novos vértices filhos, para posteriormente decidir se eles também serão ramificados.

A Figura 4.14 apresenta o algoritmo *branch-and-bound* apresentado em [55] para a solução do escalonamento de tarefas em multiprocessadores. O algoritmo recebe um grafo de programa como entrada e produz um grafo de tarefas “anotado”, contendo informações sobre os tempos de início e término das tarefas para o melhor escalonamento. Neste algoritmo, um *conjunto ativo AS* é utilizado para armazenar todos os vértices ativos.

0 algoritmo Branch-and-Bound

```
1. Inicialize o conjunto ativo AS com o vértice raiz;
2. Atribua como melhor vértice V o vértice raiz;
3. ENQUANTO {AS ≠ ∅}
4.   Selecione um vértice V1 em AS de acordo com a regra de seleção de
   vértice S
5.   SE {condição de parada para S} ENTÃO
       Vá para o passo 10
6.   Gere um conjunto DB de vértices filhos para o vértice V1 de acordo regra
   de ramificação de vértices B;
7.   Calcule o custo  $L(v)$  para cada vértice  $v$  em DB usando a função de menor-
   limite  $L$ ;
8.   Elimine vértice em DB e AS de acordo com a regra E de eliminação de
   vértices;
9.   Mova todos os vértices que permaneceram em DB para AS;
10. FIM ENQUANTO
```

Fig. 4.14: Algoritmo Branch-and-Bound.

Capítulo 5

Modelos Publicados e Experimentos Iniciais

Neste capítulo apresentamos a arquitetura do escalonador de tarefas em multiprocessadores baseado em ACs proposto em [94], os modelos de vizinhança de AC propostos na literatura [94, 102] e utilizados neste trabalho, e experimentos que reproduzimos (simulamos) a partir de publicações dos principais pesquisadores nesta área.

Este capítulo está organizado da seguinte maneira. Na Seção 5.1 são apresentados os principais conceitos, os modelos de vizinhança e a arquitetura de um escalonador de tarefas em multiprocessadores baseado em autômatos celulares (ACs), proposto originalmente em [94]. Na Seção 5.2 discutimos a capacidade de generalização das regras de transição de ACs, que é fundamental para o escalonador baseado em ACs. Na Seção 5.3 apresentamos os principais grafos de programa utilizados nos experimentos realizados. Na Seção 5.4 são apresentadas as configurações utilizadas para a realização dos experimentos. Na Seção 5.5 são apresentados resultados experimentais obtidos com a utilização do modelo de vizinhança linear de AC. Na Seção 5.6 são apresentados resultados experimentais obtidos com a utilização dos modelos de vizinhança não-lineares e com o modo de operação (atualização) de ACs seqüencial (Seção 3.2.2). Os resultados experimentais apresentados nas Seções 5.5 e 5.6 incluem alguns experimentos que simulamos, a partir de resultados originais, isto é, trabalhos relacionados publicados, buscando reproduzir e aprofundar esses resultados, além de alguns experimentos que realizamos para avaliar a capacidade de generalização das regras descobertas.

Nos experimentos descritos nesse capítulo, utilizamos a mesma abordagem proposta em [94], que chamamos de Evolução Simples. No próximo capítulo, descreveremos experimentos que realizamos

em algumas modificações da arquitetura do escalonador, que propomos nessa dissertação. Essas modificações se referem à fase de aprendizagem, onde a busca genética sofreu alterações com o objetivo de melhorar a capacidade de generalização das regras evoluídas.

5.1 Escalonamento de Tarefas em Multiprocessadores baseado em Regras de Autômatos Celulares

Nesta seção, são apresentados os principais conceitos e a arquitetura de um escalonador de tarefas em multiprocessadores baseado em autômatos celulares (ACs). Essa arquitetura foi proposta inicialmente em [94]. São apresentados três modelos diferentes para a formação da vizinhança do AC utilizado para o escalonamento: uma vizinhança linear [102], baseada no raio r do AC, e duas vizinhanças não-lineares [96], associadas a uma tarefa k do grafo de programa, baseadas nos conjuntos de tarefas *predecessoras*(k), *irmãs*(k) e *sucessoras*(k).

5.1.1 Conceito do escalonador baseado em ACs

No escalonador de tarefas em multiprocessadores baseado em ACs proposto em [94], cada célula do reticulado do AC é associada a uma tarefa do grafo de programa e o número de estados que cada célula pode assumir é associado ao número de processadores. Considerando um sistema multiprocessador composto por dois processadores (P0 e P1), o AC utilizado é binário. Cada célula do AC pode assumir, em um instante de tempo t , o valor 0, indicando que a tarefa correspondente está alocada no processador P0, ou o valor 1, indicando que a tarefa está alocada no processador P1. Por exemplo, um grafo de programa composto por 4 tarefas é representado por um reticulado de 4 células. Uma configuração onde as tarefas 0 e 3 estão alocadas em P0 e as tarefas 1 e 2 em P1, corresponde ao reticulado 0110.

O conceito do escalonador baseado em ACs é ilustrado na Figura 5.1. Inicialmente, as tarefas do grafo de programa são aleatoriamente alocadas (distribuídas) nos processadores. Uma alocação inicial das tarefas em um grafo de sistema pode ser considerada como sendo as tarefas 0 e 3 alocadas em P0, e as tarefas 1 e 2 alocadas em P1 (Figura 5.1), correspondendo à configuração inicial do AC que resulta

no reticulado 0110 (Figura 5.1). Na seqüência, o AC começa a sua evolução temporal de acordo com sua regra de transição de estados. Alterações de estado no reticulado do AC correspondem a alterações da alocação das tarefas no grafo de sistema, o que pode resultar em mudança no tempo total de execução T do programa paralelo. O estado final do AC corresponde a uma alocação final das tarefas nos processadores. Por exemplo, na Figura 5.1, o reticulado final representa uma alocação das tarefas 0 e 2 em P0 e das tarefas 1 e 3 em P1. Espera-se que essa alocação final corresponda à otimização de T .

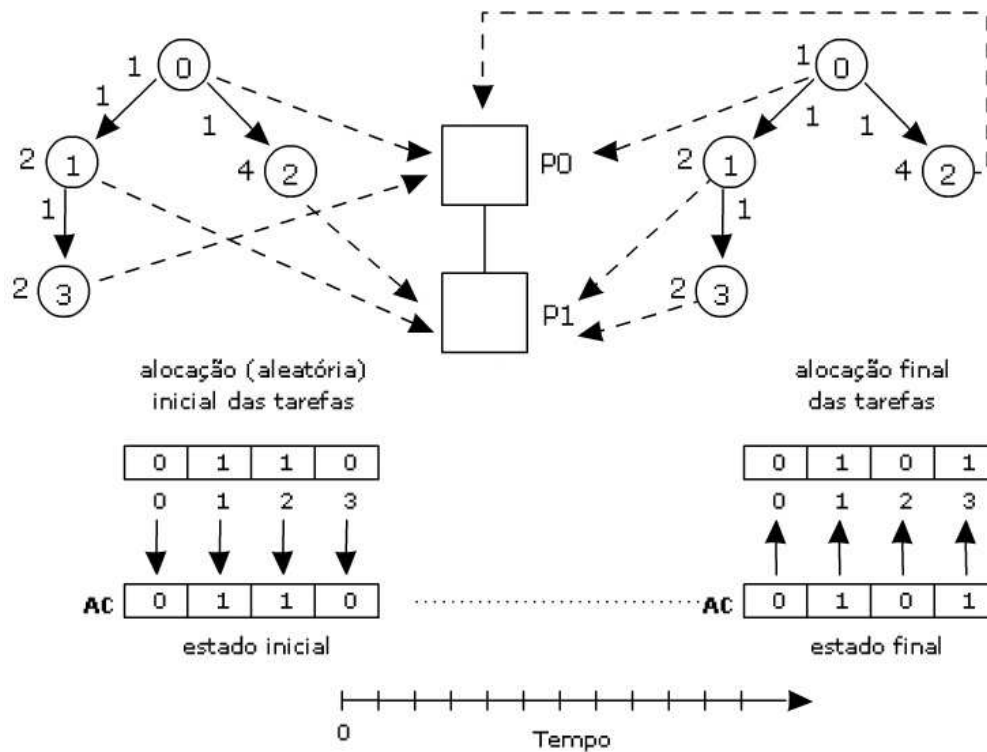


Fig. 5.1: O conceito do escalonador baseado em ACs proposto em [94].

No projeto de um escalonador baseado em ACs, devem ser respondidas duas questões iniciais [99]:

1. Que tipo de vizinhança deve ser utilizada no AC para capturar as relações entre as tarefas expressas no grafo de programa;
2. Como encontrar, em um enorme espaço de regras de transição de ACs, regras aptas a resolver o problema do escalonamento.

Nas próximas seções, serão apresentados três tipos de vizinhança, uma linear e duas outras não-lineares, comumente utilizadas no projeto de escalonadores baseados em ACs [96, 102]. Na Seção 5.1.4, a arquitetura completa do escalonador é apresentada, discutindo-se como o AG pode ser utilizado na busca de regras de ACs aptas ao escalonamento.

5.1.2 Modelo de vizinhança linear

No projeto do algoritmo de escalonamento baseado em AC com vizinhança linear [102, 104], assume-se que a estrutura não-linear de um grafo de programa composto por N_p tarefas é aproximada para a estrutura linear de um AC unidimensional com um reticulado formado por N_p células. A Figura 5.2a ilustra um reticulado de um AC unidimensional binário, composto por 18 células, que corresponde a uma alocação das tarefas do grafo de programa Gauss18 (Figura 4.2) em uma arquitetura multiprocessadora com dois processadores. Cada tarefa do Gauss18 está associada a uma célula do reticulado do AC. Ou seja, a tarefa 0 está associada à célula 0, a tarefa 1 está associada à célula 1, e assim por diante. O estado de cada célula representa o processador no qual a tarefa correspondente está alocada, sendo que 0 corresponde ao processador P0 e 1 ao processador P1. Assim, o reticulado da Figura 5.2a ilustra uma alocação do Gauss18 onde as tarefas 0, 5, 6, 8, 10, 13, 14, 15 e 16 estão alocadas no processador P0 e as demais tarefas estão alocadas no processador P1. A vizinhança correspondente para cada célula do AC é muito simples. Ela inclui a própria célula e um raio de r vizinhas de cada lado da célula. Considere, por exemplo, um raio $r = 1$. Na Figura 5.2a é apresentada a formação da vizinhança para a célula 9, associada à tarefa 9, que é formada pela célula 8 (vizinha à esquerda), a própria célula 9 e a célula 10 (vizinha à direita). Para raios maiores, a elaboração da vizinhança é feita de forma similar. No modelo descrito em [102], o AC é considerado com condição de contorno não-periódica (Seção 3.2.1), e as células “pontilhadas” são *células ausentes* utilizadas apenas para a formação da vizinhança, onde os seus estados estão sempre mapeados para o valor 0 (zero).

A Figura 5.2b apresenta um exemplo de regra de transição para um AC unidimensional binário de raio $r = 1$. A vizinhança η_i da célula 9 (Figura 5.2a) é 010, sendo que a regra de transição $\Phi(\eta_i)$ mapeia o estado da célula central (a célula 9) para o estado 0. Ou seja, no próximo estado (configuração) do AC, o estado da célula 9 será 0. Em outras palavras, a tarefa 9 está alocada no processador P1 no instante de tempo retratado na Figura 5.2a, mas será alocada no processador P0 no próximo instante

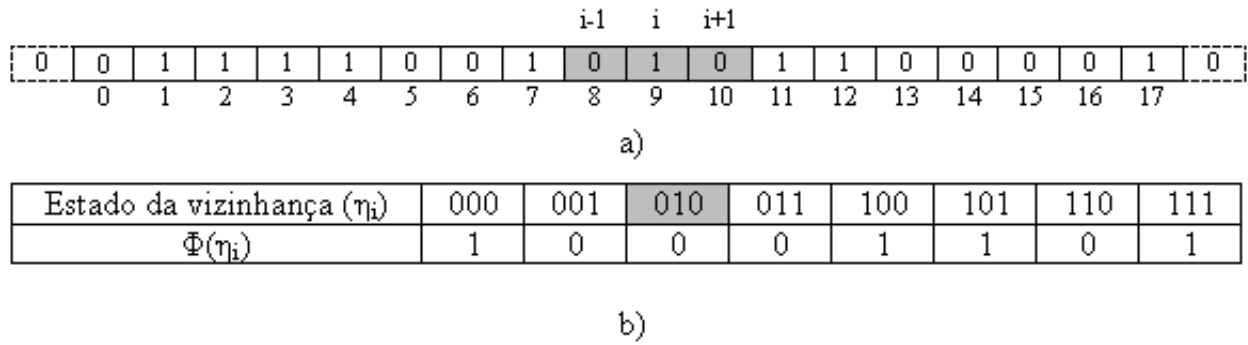


Fig. 5.2: (a) Exemplo de um reticulado de AC representando a alocação do grafo de programa Gauss18 em um sistema com dois processadores. A vizinhança é linear, de raio $r = 1$. (b) Uma regra de transição para um AC unidimensional binário de raio 1.

de tempo, devido à sua vizinhança e à regra de transição.

Considerando a vizinhança linear de um AC binário, o tamanho da regra de transição do AC é dado por $L = 2^{2r+1}$. Sendo assim, para $r = 1$ tem-se que $L = 8$, para $r = 2$ tem-se que $L = 32$ e assim por diante. Ou seja, o tamanho da regra aumenta exponencialmente em função do raio escolhido.

5.1.3 Modelos de vizinhança não-lineares

Em [96] são apresentados dois diferentes modelos para a formação de vizinhanças de ACs onde a estrutura não-linear de um grafo de programa é modelada através de uma estrutura também não-linear de vizinhança de um AC. Em ambos modelos não-lineares, assim como no modelo linear discutido na seção anterior, cada célula do reticulado é associada a uma tarefa do programa paralelo. A alteração se refere à forma como a vizinhança de uma célula é especificada. Nos modelos não-lineares, a simples proximidade de duas tarefas no reticulado não as tornam vizinhas.

Nestes modelos, a vizinhança de uma célula do AC, associada a uma tarefa do grafo de programa, é baseada no conjunto de tarefas *predecessoras*, *irmãs* e *sucessoras* da tarefa analisada, discutidos na Seção 4.1. A Figura 5.3 apresenta um reticulado de 18 células associado ao Gauss18 e destaca as tarefas que fazem parte da vizinhança da tarefa 8. Essa vizinhança é formada pelos conjuntos $predecessoras(8) = \{3,6\}$, $irmãs(8) = \{7,9,10,11\}$ e $sucessoras(8) = \{11,12\}$, e a própria célula 8.

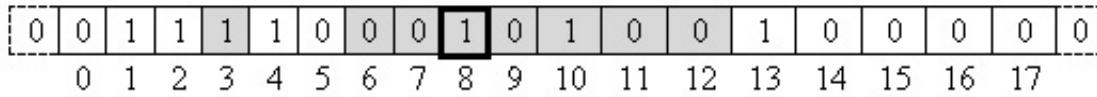


Fig. 5.3: Exemplo de reticulado de 18 células associado ao Gauss18, destacando-se as tarefas que fazem parte da vizinhança da tarefa 8.

Vizinhança Seleccionada

A *vizinhança seleccionada* de uma tarefa k do grafo de programa é formada pela própria tarefa e por somente duas tarefas representativas de cada conjunto de tarefas predecessoras, irmãs e sucessoras da tarefa k . Estas tarefas representativas são, respectivamente, a tarefa com o valor máximo e a tarefa com o valor mínimo para algum atributo das tarefas escolhido *a priori*. Os possíveis atributos para uma tarefa k do grafo de programa são: custo computacional, custo de comunicação, nível estático, nível dinâmico, co-nível estático e co-nível dinâmico. Em uma dada execução do algoritmo de escalonamento, um atributo para cada conjunto de tarefas predecessoras, irmãs e sucessoras é seleccionado, podendo estes atributos serem diferentes para cada conjunto. Por exemplo, em uma execução do algoritmo, os atributos seleccionados para o conjunto de predecessoras, irmãs e sucessoras podem ser, respectivamente, o co-nível estático, o nível estático e o custo de comunicação.

Este modelo de vizinhança utiliza 7 células (incluindo a célula k). As tarefas representativas associadas a cada conjunto de predecessoras, irmãs e sucessoras serão referenciadas como uma *subvizinhança* de k . Assim, a vizinhança de uma célula k é formada pela própria célula k , pela subvizinhança de predecessoras, pela subvizinhança de irmãs e pela subvizinhança de sucessoras. Se as tarefas correspondentes de uma subvizinhança estão “ausentes” no grafo de programa (ou seja, o conjunto associado à subvizinhança é vazio), tarefas “postiças” são introduzidas e, como veremos adiante, receberão um tratamento especial.

Como a estrutura de um grafo de programa é irregular, o número de predecessoras, irmãs ou sucessoras pode ser menor que dois ou elas podem ter os mesmos valores para os atributos. As seguintes soluções para casos especiais são adotadas:

- Se predecessoras (irmãs ou sucessoras) não existem para uma dada tarefa, uma subvizinhança correspondente para cada situação é criada adicionando-se um par de tarefas “postiças” e associando-se estas tarefas com um par de células; os estados destas células (representando

processadores onde as tarefas estão alocadas) são indefinidos e o estado de cada subvizinhança recebe um valor especial.

- Se apenas uma predecessora (irmã ou sucessora) existe para uma dada tarefa, adiciona-se apenas uma tarefa “postiça” para a subvizinhança correspondente e o estado da nova célula receberá o mesmo estado da tarefa existente. Ou seja, a tarefa postiça e a tarefa real existente para uma dada subvizinhança estarão alocadas no mesmo processador.
- Se o número de predecessoras (irmãs ou sucessoras) for maior que dois e todas elas tiverem o mesmo valor para um dado atributo, serão selecionadas as duas tarefas com o menor e maior número de ordem.

A Figura 5.4 ilustra a criação da vizinhança para a tarefa 8 do Gauss18, onde foram considerados os atributos co-nível estático (d_k), custo computacional (b_k) e custo de comunicação (a_{kl}) para os conjuntos de predecessoras, irmãs e sucessoras, respectivamente. Temos que predecessoras(8)={3,6}, irmãs(8)={7,9,10,11} e sucessoras(8)={11,12}, e os valores dos atributos selecionados para cada conjunto estão identificados na Figura 5.4, para cada tarefa pertencente ao respectivo conjunto.

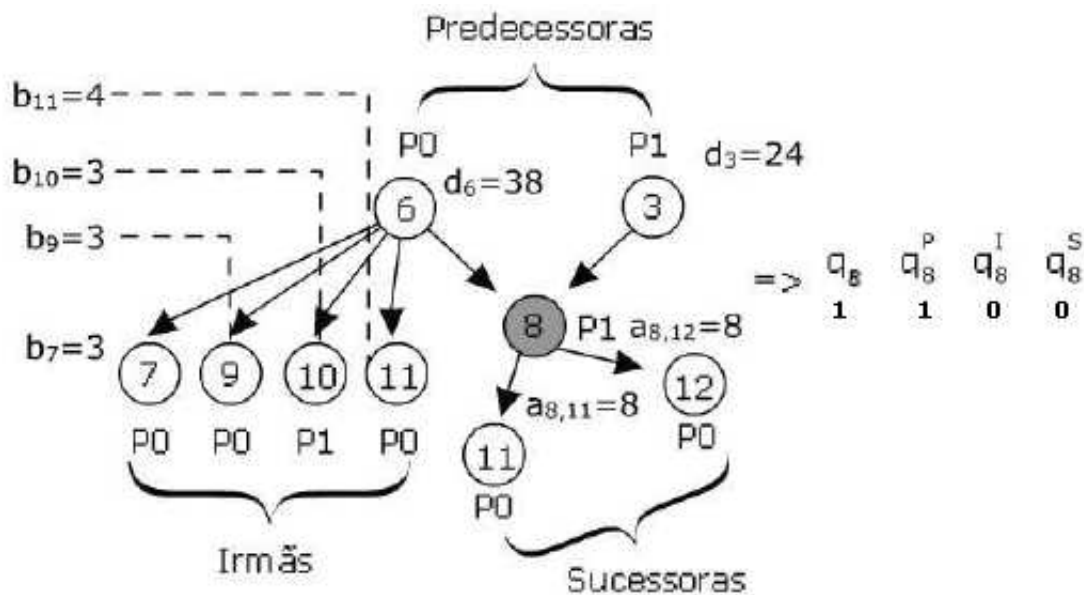


Fig. 5.4: Vizinhança selecionada para a tarefa 8 do Gauss18.

Construída a vizinhança, é necessário definir estados das subvizinhanças q_k^P (predecessoras de k),

q_k^I (irmãs de k), q_k^S (sucessoras de k) e o estado q'_k , que representa o novo estado da tarefa k . O estado da célula k (q_k) assume o valor 0 ou 1, dependendo se a tarefa k está alocada no processador P0 ou P1, respectivamente. Os valores de cada par de células correspondentes às subvizinhanças são mapeados em um de cinco valores descrevendo o estado do par da seguinte maneira:

- estado 0: os valores das células do par são iguais a 0 (ambas as tarefas estão alocadas em P0).
- estado 1: a primeira célula tem o valor 0 e a segunda célula tem o valor 1 (a tarefa com o maior valor do atributo está alocada em P0 e a tarefa com o menor valor está em P1).
- estado 2: a primeira célula tem o valor 1 e a segunda célula tem o valor 0 (a tarefa com o maior valor do atributo está alocada em P1 e a tarefa com o menor valor está em P0).
- estado 3: os valores das células do par são iguais a 1 (ambas as tarefas estão alocadas em P1).
- estado 4: os valores das duas células são indefinidos (não há tarefas correspondentes para estas células). Ou seja, foi inserido no grafo de programa um par de tarefas “postiças”.

O co-nível das tarefas 3 e 6, que são predecessoras da tarefa 8, é 24 e 38, respectivamente. Como mostra a Figura 5.4, a tarefa 3 está alocada em P1 e a tarefa 6 está alocada em P0. De acordo com a regra acima, o estado de $q_8^P = 1$. Para o conjunto das tarefas irmãs da tarefa 8, a tarefa 11 é a que possui o maior valor para o atributo custo computacional ($b_{11} = 4$). As outras três tarefas irmãs (7, 9 e 10) possuem o mesmo valor para este atributo e, de acordo com o mencionado como soluções para casos especiais, a tarefa selecionada deve ser a de menor número de ordem, isto é, a tarefa 7. Como ambas as tarefas estão alocadas no processador P0, $q_8^I = 0$. No caso das tarefas sucessoras (11 e 12), existem apenas duas e ambas estão alocadas no processador P0. Então, $q_8^S = 0$. A tarefa está alocada em P1, logo $q_8 = 1$. Assim, a vizinhança da tarefa 8 é dada por: $q_8 = 1$, $q_8^P = 1$, $q_8^I = 0$ e $q_8^S = 0$. A regra de transição deve definir para cada vizinhança $(q_k, q_k^P, q_k^I, q_k^S)$ qual será o novo estado q'_k da célula k :

$$(q_k, q_k^P, q_k^I, q_k^S) \rightarrow q'_k \quad (5.1)$$

Como o estado q_k da célula central pode assumir um de dois valores ($\{0,1\}$), e os estados q_k^P, q_k^I, q_k^S podem assumir um de cinco valores ($\{0,1,2,3,4\}$), o número total de vizinhanças possíveis é $2 \times 5 \times$

$5 \times 5 = 250$ [96]. O comprimento de uma regra de transição é 250 bits, e então o espaço de regras é formado por 2^{250} regras.

Vizinhança Totalística

Este modelo foi proposto visando reduzir o tamanho da vizinhança do AC e, conseqüentemente, reduzir o tamanho da regra de transição [96]. Diferentemente da vizinhança selecionada, este modelo considera todas as tarefas pertencentes aos conjuntos de predecessoras, irmãs e sucessoras de uma dada tarefa central, e não apenas as mais representativas (a maior e a menor) de cada conjunto.

A vizinhança de uma célula k é constituída pela própria célula (q_k) e mais outras três: a primeira representando o conjunto de predecessoras de k (q_k^P), a segunda representando o conjunto de irmãs de k (q_k^I) e a terceira representando o conjunto de sucessoras de k (q_k^S). Os mesmos atributos do grafo de programa que podem ser utilizados na formação da vizinhança selecionada também podem ser utilizados na vizinhança totalística. O estado da célula associada a um dado conjunto é definido de forma similar à definição de regras totalísticas em ACs lineares (Seção 3.3.1) [96].

Consideremos, por exemplo, o cálculo do estado da célula q_k^P associada ao conjunto de predecessoras de k . P_{P0}^k e P_{P1}^k são os conjuntos de predecessoras da tarefa k alocadas em P0 e P1, respectivamente. Considere que o atributo selecionado para os predecessores é o co-nível estático (d). O estado da célula q_k^P é definido conforme a seguir, sendo $i \in P_{P0}^k$ e $j \in P_{P1}^k$:

- $q_k^P = 0$, se $\sum_i d_i > \sum_j d_j$
- $q_k^P = 1$, se $\sum_i d_i < \sum_j d_j$
- $q_k^P = 2$, se $\sum_i d_i = \sum_j d_j$
- $q_k^P = 3$, se não há predecessoras

O cálculo dos estados das células associadas ao conjunto de irmãs (q_k^I) e ao conjunto de sucessoras (q_k^S) é realizado de forma análoga, utilizando-se os conjuntos I_{P0}^k (irmãs de k alocadas em P0), I_{P1}^k (irmãs de k alocadas em P1), S_{P0}^k (sucessoras de k alocadas em P0) e S_{P1}^k (sucessoras de k alocadas em P1). Para cada tipo de conjunto pode ser associado um atributo diferente.

A Figura 5.5 ilustra a criação da vizinhança totalística para a tarefa 8 do Gauss18, onde foram considerados os atributos co-nível estático (d_k), custo computacional (b_k) e custo de comunicação

(a_{kl}) para os conjuntos de predecessoras, irmãs e sucessoras, respectivamente. O co-nível das tarefas 3 e 6, que são predecessoras da tarefa 8, é 24 e 38, respectivamente. Como mostra a Figura 5.5, a tarefa 3 está alocada no processador P1 e a tarefa 6 está alocada no processador P0, ou seja, $P_{P0}^8 = \{6\}$ e $P_{P1}^8 = \{3\}$. Logo, $\sum_{i \in P_{P0}^8} d_i = 38$, $\sum_{j \in P_{P1}^8} d_j = 24$ e é assumido $q_8^P = 0$. Além disso, tem-se que $I_{P0}^8 = \{7,9,11\}$, $I_{P1}^8 = \{10\}$, $S_{P0}^8 = \{11,12\}$ e $S_{P1}^8 = \{\}$. Assim, calcula-se $\sum_{i \in I_{P0}^8} b_i = 10$, $\sum_{j \in I_{P1}^8} b_j = 3$, $\sum_{i \in S_{P0}^8} a_{8i} = 16$ e $\sum_{j \in S_{P1}^8} a_{8j} = 0$. Logo, o estado correspondente tanto para as irmãs (q_8^I) quanto para as sucessoras (q_8^S) é 0. Visto que a tarefa 8 está alocada em P1, temos $q_8 = 1$. Assim, a vizinhança da tarefa 8 é dada por: $q_8 = 1$, $q_8^P = 0$, $q_8^I = 0$ e $q_8^S = 0$. A regra de transição deve definir para cada vizinhança $(q_k, q_k^P, q_k^I, q_k^S)$ qual será o novo estado q'_k da célula k (Equação 5.1).

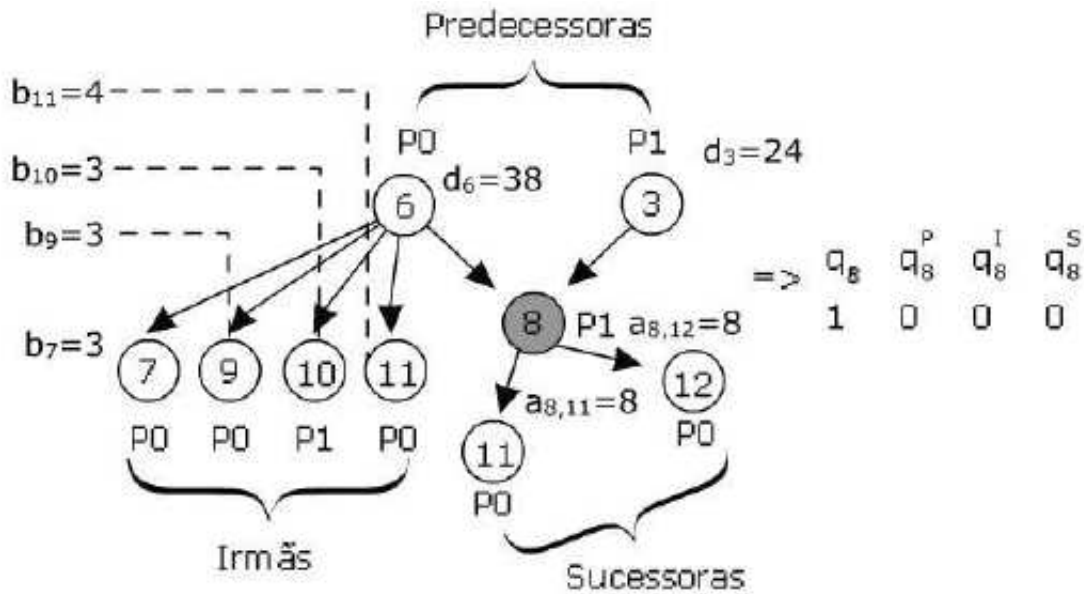


Fig. 5.5: Vizinhança totalística para a tarefa 8 do Gauss18.

Como o estado q_k da célula central pode assumir um de dois valores ($\{0,1\}$), e os estados q_k^P, q_k^I, q_k^S podem assumir um de quatro valores ($\{0,1,2,3\}$), o número total de vizinhanças diferentes é $2 \times 4 \times 4 \times 4 = 128$ [96]. O comprimento de uma regra de transição é 128 bits, e então o espaço de regras é formado por 2^{128} regras.

5.1.4 Arquitetura do escalonador baseado em ACs

Um autômato celular evolui seu reticulado de acordo com a sua regra de transição de estados. O objetivo do escalonamento baseado em ACs é encontrar regras de transição capazes de convergir qualquer configuração inicial do AC, que representa uma alocação inicial das tarefas de um grafo de programa em um programa paralelo, para uma configuração que minimize o tempo total de execução T . O algoritmo de escalonamento em multiprocessadores baseado em ACs envolve duas fases [96]: a fase de aprendizagem de regras de ACs e a fase de operação. A Figura 5.6 apresenta a arquitetura do escalonador baseado em ACs proposto em [94].

O propósito da fase de aprendizagem é descobrir regras de ACs capazes de escalonar um grafo de programa específico. Espera-se que as regras encontradas sejam aptas a resolver o problema do escalonamento para qualquer alocação inicial das tarefas para uma dada instância do problema. A busca por estas regras é feita com a utilização de um algoritmo genético (AG) [43]. O AG manipula uma população P de regras de transição de ACs, cada uma codificada como uma *palavra* binária, independentemente do modelo de vizinhança utilizado (linear, selecionada ou totalística). O cálculo da aptidão de cada regra é estimado por:

1. são geradas alocações iniciais aleatórias das tarefas (CIs - configurações iniciais);
2. aplica-se então a regra em cada configuração inicial por M passos de tempo e calcula-se T para cada alocação final;
3. o valor da aptidão da regra é o somatório dos valores de T obtidos para todas as configurações iniciais avaliadas.

Como o objetivo do escalonamento de tarefas em multiprocessadores é a minimização do tempo total de execução T , quanto menor o valor de T , melhor é a regra de transição evoluída.

Os passos básicos de execução do algoritmo genético são:

1. sorteio aleatório da população inicial de P regras;
2. sorteio de uma amostra de configurações iniciais (CIs) do reticulado, diferentes a cada geração;

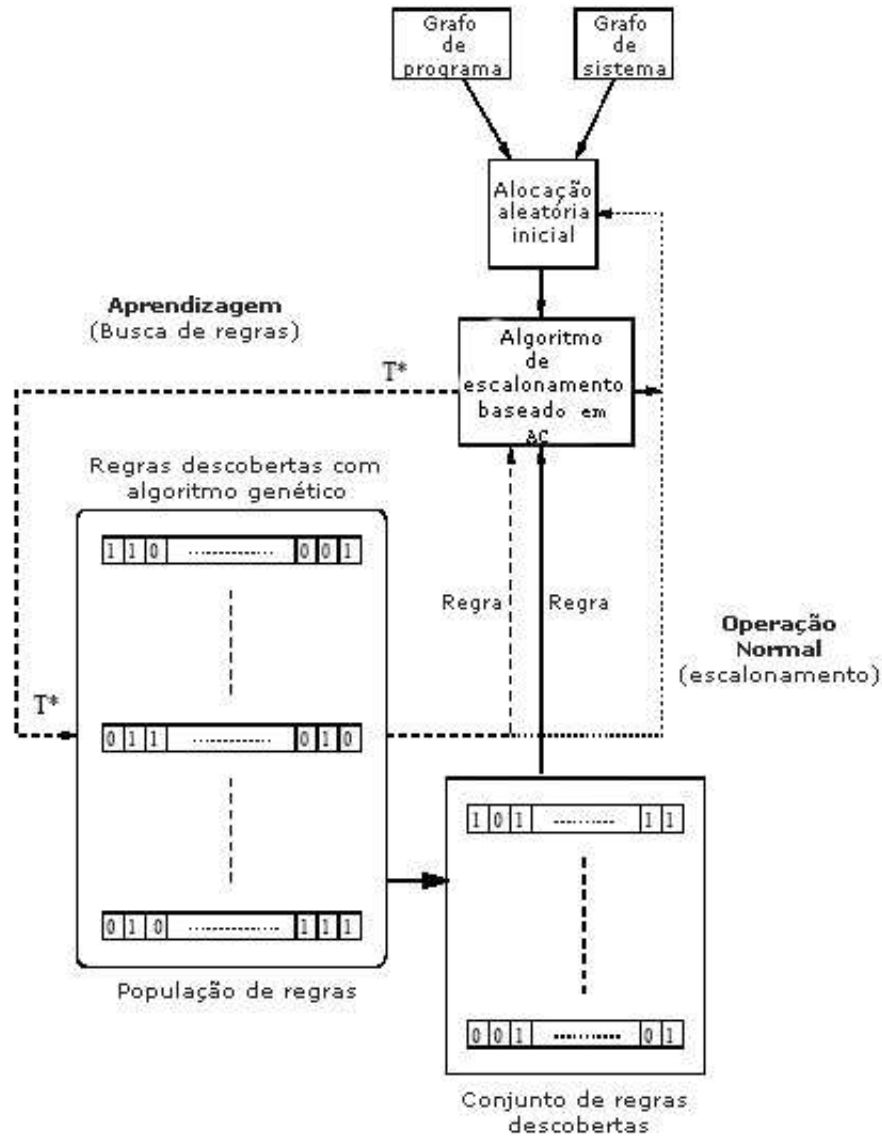


Fig. 5.6: Uma arquitetura de escalonador baseado em AC [96].

3. cada regra da população é avaliada na amostra de CIs, calculando-se o valor de T , conforme descrito anteriormente (cálculo da aptidão);
4. as E melhores regras (Elite) são copiadas para a próxima geração sem nenhuma alteração;
5. as $P - E$ regras restantes são formadas aplicando-se o operador *crossover* de um ponto para os pares de regras da elite selecionados aleatoriamente;
6. as regras “filhas”, geradas a cada *crossover*, são sujeitas à aplicação do operador de mutação

com probabilidade p_m ;

7. o processo volta ao passo 2 até que o número máximo de gerações G seja alcançado;
8. a população da última geração é armazenada em um repositório de regras.

Na fase de operação, o grafo de programa usado na fase de aprendizagem é aleatoriamente alocado, e o AC é inicializado e equipado com uma regra obtida do conjunto de regras descobertas na fase anterior. Espera-se que, para qualquer alocação inicial das tarefas, a regra do AC seja capaz de encontrar, em um número finito de passos, uma alocação das tarefas que provê o valor ótimo (ou próximo) de T .

Em todos os resultados experimentais apresentados nessa dissertação, o cálculo da aptidão de uma regra de transição de AC utilizado foi o descrito nesta seção. E, na maioria dos experimentos descritos nesse capítulo, os passos de execução do algoritmo genético (AG) utilizados também foram os descritos aqui.

Além de se esperar que as regras encontradas na fase de aprendizagem sejam capazes de encontrar uma alocação das tarefas que provê o valor ótimo (ou próximo) de T para um determinado grafo de programa, espera-se que elas também possam ser utilizadas com um bom desempenho em diferentes grafos de programa. Isto é caracterizado como a *capacidade de generalização* das regras, que será o foco da próxima seção.

5.2 Capacidade de generalização das regras de ACs

A capacidade de generalização de uma regra de um AC é medida com base no desempenho da mesma ao resolver o problema do escalonamento para diferentes grafos de programa. Ou seja, quanto mais grafos a regra conseguir escalonar com eficiência, mais geral ela é.

Em trabalhos anteriores [96, 99, 104], no algoritmo de escalonamento baseado em ACs, foi utilizado apenas um grafo na evolução das regras (primeira fase). Ou seja, o AG evolui as regras calculando a aptidão das mesmas em um único grafo. Resultados publicados mostram que o algoritmo de escalonamento baseado em ACs foi capaz de descobrir regras para o escalonamento ótimo de vários programas paralelos conhecidos na literatura [96, 97, 99, 104]. Entretanto, quando as regras

encontradas para um determinado grafo eram aplicadas a outros programas, resultados conclusivos sobre a capacidade de generalização das regras não foram apresentados. Os trabalhos publicados são promissores, mas ainda não apresentaram o desempenho desejado.

É interessante ressaltarmos que essa capacidade de generalização das regras evoluídas na fase de aprendizagem é crucial para o escalonamento baseado em ACs. Todo o esforço computacional empregado na descoberta das regras de ACs só se justifica se elas puderem ser reutilizadas em novos problemas. Do contrário, um AG simples pode ser utilizado diretamente na busca da configuração ótima de cada grafo independentemente, sem a necessidade de envolvermos os ACs no modelo [46, 53]. A idéia de se utilizar as regras de ACs é justamente pela possibilidade de reaproveitamento das mesmas em novos programas paralelos, sem a necessidade de um novo processo de aprendizagem evolutivo, que exige um grande esforço computacional.

5.3 Grafos de programa utilizados

Os grafos de programa mais utilizados nos experimentos descritos nas próximas seções foram o Gauss18 (Figura 4.2), o *g18* (Figura 5.7a), o *tree15* (Figura 5.7b), o *g40* (Figura 5.7c) e o *intree15* (Figura 5.7d). O *g18*, assim como o Gauss18, é um grafo de programa composto por 18 tarefas. Porém, sua estrutura é mais regular que a estrutura do Gauss18, visto que ele não apresenta, por exemplo, situações onde uma tarefa é, ao mesmo tempo, sucessora e irmã de uma outra. No *g18*, todos os custos de comunicação são iguais a 1 e o custo computacional de cada tarefa é apresentado na figura. O *g40* é um grafo de programa composto por 40 tarefas, também em uma estrutura regular, onde todas as tarefas possuem custo computacional igual a 4 e todos os custos de comunicação são iguais a 1. O *tree15* e o *intree15* são grafos de programa em formato de árvore binária, compostos por 15 tarefas, onde todos os custos computacionais e de comunicação são iguais a 1. A Tabela 5.1 apresenta o tempo total de execução mínimo (escalonamento ótimo) para cada grafo de programa alocado em um sistema com dois processadores [97, 102, 103, 104].

Para representarmos a estrutura de um grafo de programa, utilizaremos a notação comum de grafos e seus conjuntos de vértices (V) e arestas (E). Por exemplo, a representação do Gauss18 utilizando-se esta notação é:

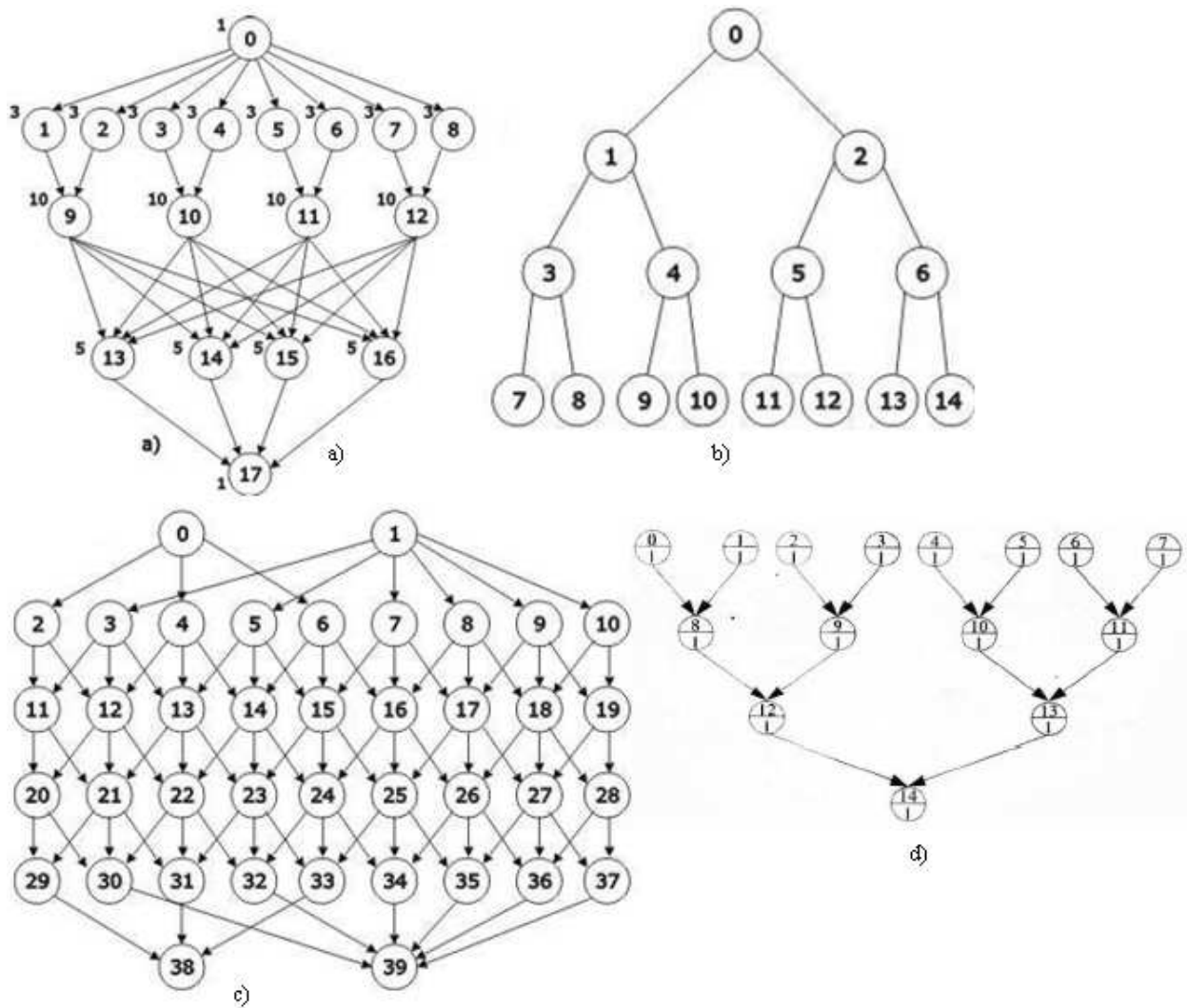


Fig. 5.7: Grafos de programa: (a) *g18*: custo de comunicação igual a 1, (b) *tree15*: custos de comunicação e computacional igual a 1, (c) *g40*: custo de comunicação igual a 1 e custo computacional igual a 4 e (d) *intree15*: custos de comunicação e computacional igual a 1.

$V = \{(0,8), (1,4), (2,4), (3,4), (4,4), (5,4), (6,6), (7,3), (8,3), (9,3), (10,3), (11,4), (12,2), (13,2), (14,2), (15,2), (16,1), (17,1)\}$ e

$E = \{(0,1,12), (0,2,12), (0,3,12), (0,4,12), (0,5,12), (0,6,12), (2,6,8), (2,7,8), (3,8,8), (4,9,8), (5,10,8), (6,7,12), (6,8,12), (6,9,12), (6,10,12), (6,11,12), (8,11,8), (8,12,8), (9,13,8), (10,14,8), (11,12,12), (11,13,12), (11,14,12), (11,15,12), (13,15,8), (13,16,8), (14,17,8), (15,16,12), (15,17,12)\}$

sendo que cada elemento $(k, b_k) \in V$ representa a tarefa k e seu custo computacional b_k , e cada elemento $(k, l, a_{kl}) \in E$ representa que a tarefa k precede a tarefa l e o custo de comunicação entre elas é

a_{kl} .

Grafo de Programa	T_{min}
Gauss18	44,0
<i>g18</i>	46,0
<i>g40</i>	80,0
<i>tree15</i>	9,0
<i>intree15</i>	9,0

Tab. 5.1: Grafos de programa e seus tempos de execução T mínimos em um sistema com dois processadores.

Em alguns experimentos apresentados neste capítulo e no Capítulo 6, foram utilizados quinze grafos de programa representando variações do Gauss18. Estas variações são grafos semelhantes ao Gauss18, porém apresentando modificações nos custos computacionais de algumas tarefas, modificações nos custos de comunicação entre as tarefas, inserção e exclusão de arestas (*links*) entre as tarefas. Cada variação do Gauss18 deve apresentar, no mínimo, uma das modificações acima citadas, podendo apresentar mais de uma ou todas elas. O primeiro conjunto de variações do Gauss18 utilizado nos experimentos possui quinze grafos que estão apresentados na Tabela 5.2, utilizando-se a notação descrita, onde V e E são os conjuntos de vértices e arestas originais do Gauss18, respectivamente.

Para estimar o valor de T mínimo para cada variação do Gauss18, foram utilizados dois modelos de AG para escalonar tarefas. O propósito destes AGs é encontrar o escalonamento diretamente, sem o uso de regras de transição de ACs. Na Seção 4.3.1, discutimos o modelo de AG apresentado em [89] para o escalonamento de tarefas em um sistema com dois processadores, sem envolver os ACs. Nesse modelo, os indivíduos da população representam a distribuição das tarefas entre os processadores e, além disso, a ordem em que as tarefas deverão ser executadas em cada um destes processadores. Uma implementação reproduzindo o modelo do AG descrito em [89] foi feita por Lucas Bucci da Silveira, aluno de mestrado e colega de pesquisa da Universidade Federal de Uberlândia. Esse modelo foi chamado de *AG1* e os resultados obtidos com a utilização do *AG1* nas quinze variações do Gauss18 estão listados na segunda coluna da Tabela 5.3. Estes resultados apresentam os melhores tempos obtidos em 30 execuções do *AG1* e servem como valores de referência para as variações do Gauss18. Um segundo modelo foi implementado no qual o AG encontra somente a alocação (distribuição) das tarefas entre os processadores, enquanto a ordem de execução das tarefas dentro de cada processador foi baseada em uma política de escalonamento fixa. A política de escalonamento utilizada foi a

mesma utilizada na maioria dos experimentos com o modelo de escalonador baseado em AC: “a tarefa com maior nível dinâmico é executada primeiro”. Esse segundo modelo foi chamado AG2. Como nesse modelo o indivíduo representa apenas a distribuição das tarefas entre os processadores, sendo a ordem de execução das tarefas em cada processador determinada por uma política de escalonamento e não embutida no próprio indivíduo, acreditamos poder nos referenciar, com maior garantia, nos tempos que podem ser obtidos para a variações do Gauss18. Desta forma, os resultados obtidos com o AG2 são mais próximos dos possíveis de serem obtidos com o algoritmo de escalonamento baseado em AC, onde o cálculo do tempo total de execução T para um dado grafo de programa é realizado em função de uma alocação das tarefas nos processadores e da mesma política de escalonamento.

Grafo de Programa	Representação
Gauss18_1	$V1 = V - \{(6,6)\} + \{(6,10)\}$, $E1 = E - \{(11,13,12),(11,14,12)\} + \{(11,14,4),(5,13,12)\}$
Gauss18_2	$V2 = V - \{(6,6)\} + \{(6,10)\}$, $E2 = E - \{(3,8,8),(11,13,12)\} + \{(3,6,8),(5,13,12)\}$
Gauss18_3	$V3 = V$, $E3 = E - \{(3,8,8),(11,15,12)\} + \{(3,6,8),(10,13,12)\}$
Gauss18_4	$V4 = V$, $E4 = E - \{(3,8,8),(11,15,12),(5,10,8)\} + \{(3,6,8),(10,13,12),(5,10,15)\}$
Gauss18_5	$V5 = V - \{(9,3)\} + \{(9,6)\}$, $E5 = E - \{(3,8,8),(11,15,12),(5,10,8)\} + \{(3,6,8),(10,13,12),(5,10,15)\}$
Gauss18_6	$V6 = V - \{(6,6)\} + \{(6,10)\}$, $E6 = E$
Gauss18_7	$V7 = V$, $E7 = E - \{(11,14,12)\} + \{(11,14,4)\}$
Gauss18_8	$V8 = V$, $E8 = E - \{(11,13,12)\} + \{(5,13,12)\}$
Gauss18_9	$V9 = V$, $E9 = E - \{(3,8,8)\} + \{(3,6,8)\}$
Gauss18_10	$V10 = V$, $E10 = E - \{(11,15,12)\}$
Gauss18_11	$V11 = V$, $E11 = E + \{(10,13,12)\}$
Gauss18_12	$V12 = V - \{(6,6)\} + \{(6,10)\}$, $E12 = E - \{(11,14,12)\} + \{(11,14,4)\}$
Gauss18_13	$V13 = V$, $E13 = E - \{(11,13,12),(11,14,12)\} + \{(11,14,4),(5,13,12)\}$
Gauss18_14	$V14 = V$, $E14 = E - \{(3,8,8),(11,13,12)\} + \{(3,6,8),(5,13,12)\}$
Gauss18_15	$V15 = V$, $E15 = E - \{(3,8,8),(11,15,12)\} + \{(3,6,8)\}$

Tab. 5.2: Variações do Gauss18

Trinta execuções do AG2 foram conduzidas para cada variação do Gauss18 e os melhores tempos obtidos são apresentados na última coluna da Tabela 5.3. Embora os melhores resultados encontrados pelos AGs não possam ser assumidos como os valores ótimos para cada grafo de programa, eles podem ser usados como valores de referência de bons resultados a serem alcançados pelas regras de transição de ACs. Outros grafos de programa, gerados aleatoriamente, foram utilizados em alguns experimentos. Quando mencionarmos estes experimentos descreveremos, sucintamente, a estrutura destes grafos.

Grafo	AG1 T_{min}	AG2 T_{min}
Gauss18_1	48,0	47,0
Gauss18_2	47,0	47,0
Gauss18_3	46,0	47,0
Gauss18_4	47,0	47,0
Gauss18_5	50,0	50,0
Gauss18_6	47,0	47,0
Gauss18_7	44,0	44,0
Gauss18_8	44,0	44,0
Gauss18_9	44,0	47,0
Gauss18_10	44,0	44,0
Gauss18_11	46,0	47,0
Gauss18_12	47,0	47,0
Gauss18_13	44,0	44,0
Gauss18_14	46,0	46,0
Gauss18_15	46,0	47,0
Média	46,00	46,33

Tab. 5.3: Experimento de referência para determinação dos valores de T mínimos para as quinze variações do Gauss18.

5.4 Sobre os experimentos

Nos experimentos descritos nas próximas seções e no Capítulo 6, foram utilizados vários grafos de programa como teste para modelos de AC com diferentes modos de operação (atualização) e modelos de vizinhança. O grafo de sistema utilizado em todos esses experimentos, que representa o sistema multiprocessador, é composto por dois processadores idênticos, chamados de P0 e P1. Sendo assim, o AC utilizado para representar o problema do escalonamento de tarefas é binário. Conforme descrito na fase de aprendizagem, a busca por estas regras é conduzida com a utilização de um algoritmo genético padrão (AG). Em todos os AGs utilizados nos experimentos relatados neste capítulo, a função de avaliação (aptidão) utilizada foi a descrita na Seção 5.1.4. Em todos os experimentos relatados nesse capítulo, o algoritmo de busca utilizado segue os passos básicos do AG descrito na Seção 5.1.4.

Três modelos de vizinhança de ACs foram utilizados nos experimentos: a Vizinhança Linear proposta em [102] e descrita na Seção 5.1.2, a *Vizinhança Seleccionada* e a *Vizinhança Totalística*, ambas propostas em [94] e descritas na Seção 5.1.3. Em todos os experimentos descritos, serão especificados os parâmetros do algoritmo genético (AG) e do autômato celular (AC) utilizados nas

execuções do algoritmo de escalonamento. Estes parâmetros são listados a seguir.

- P é o tamanho da população de indivíduos do AG, ou seja, o número de regras de transição de ACs evoluídas a cada geração;
- E é o tamanho da elite (melhores indivíduos da população);
- CIs é quantidade de configurações iniciais utilizadas na avaliação das regras de transição de ACs. As CIs, também chamadas de *testes de problema*, são diferentes a cada geração;
- p_c é a probabilidade de *crossover* (cruzamento);
- p_m é a probabilidade de mutação;
- M é o número de passos na evolução temporal do AC, isto é, quantas vezes a regra de transição será, sucessivamente, aplicada ao reticulado;
- G é a quantidade de gerações, que representa a condição de parada do AG.

O método de *crossover* utilizado foi o ponto-simples e o método de mutação foi o de complemento do alelo. Optamos por estes métodos por serem de simples implementação, de baixo custo computacional e por terem apresentado bons resultados em trabalhos anteriores [94, 96, 102, 104].

Nas próximas seções (5.5 e 5.6) descreveremos os experimentos iniciais que realizamos nessa dissertação, buscando reproduzir os resultados originais. Chamamos esses resultados de "experimentos reproduzidos". Em alguns casos, realizamos também experimentos adicionais, que chamamos de "experimentos novos", com o objetivo de esclarecer alguns pontos não apresentados nos artigos originais. Em todos os experimentos descritos nesse capítulo, a política de escalonamento (Seção 4.1) adotada, que determina a ordem de execução das tarefas em cada processador, foi "a tarefa com maior nível dinâmico é executada primeiro" [96]. Os principais resultados publicados por Seredynski e seus colegas, que chamamos de "experimentos originais", com a utilização do escalonador descrito na Seção 5.1.4 em diferentes grafos, são descritos no Apêndice B.

5.5 Resultados com o modelo de vizinhança linear

5.5.1 Experimentos iniciais

Nesta seção, apresentamos alguns experimentos onde buscamos reproduzir os resultados publicados, utilizando-se o modelo de vizinhança linear. Os principais grafos de programa utilizados foram o Gauss18, *g18*, *g40*, *tree15* e o *intree15*. Nos experimentos aqui descritos, foram feitos testes utilizando-se os três modos de operação de AC (paralelo, seqüencial e aleatório) e valores para o raio r igual a 1, 2 e 3. Em todos os experimentos, o tipo de *crossover* utilizado foi o de ponto-simples e o tipo de mutação foi o de complemento do alelo.

Os primeiros experimentos que simulamos foram realizados para o Gauss18 que tem $T_{ótimo} = 44,0$. Para o AC, utilizamos os três valores de raio, os três modos de operação acima citados, e também os parâmetros $CI_s = 50$ e $M = 50$. Para o raio 1, os parâmetros que utilizamos para o AG foram $P = 50$, $E = 25$, $p_m = 0,375$ e $G = 50$. Para o raio 2 utilizamos $P = 100$, $E = 50$, $p_m = 0,093$ e $G = 100$. E, para o raio 3 utilizamos $P = 200$, $E = 100$, $p_m = 0,023$ e $G = 1.000$. Para os três valores de raio utilizamos o parâmetro $p_c = 0,50$.

A Figura 5.8 apresenta a evolução do melhor indivíduo/regra da população na fase de aprendizagem de regras do algoritmo de escalonamento baseado em AC, utilizando-se o raio r igual a 1. Como pode ser observado, o modo de operação seqüencial foi o que apresentou um melhor resultado, onde a média de T para o melhor indivíduo na última geração foi igual a 54,06. No modo aleatório, a média de T para o melhor indivíduo na última geração foi igual a 56,22 e no modo paralelo foi igual a 57,28.

A Figura 5.9 apresenta a evolução do melhor indivíduo da população na fase de aprendizagem, utilizando-se regras de raio 2. Como pode ser observado, o modo de operação seqüencial foi o que apresentou um melhor resultado, onde a média de T para o melhor indivíduo na última geração foi igual a 47,26. No modo paralelo, a média de T foi igual a 51,24 e no modo aleatório foi igual a 52,24. Comparando-se os gráficos das figuras 5.8 e 5.9, podemos observar que os experimentos com os três modos de operação retornaram resultados superiores com o aumento do raio de 1 para 2.

O último experimento foi realizado utilizando-se regras de raio 3, o qual apresentou o melhor resultado para o Gauss18. A Figura 5.10 apresenta a evolução do melhor indivíduo para o modo de operação seqüencial. Até aqui apresentamos os resultados com os três valores de raio e os três modos

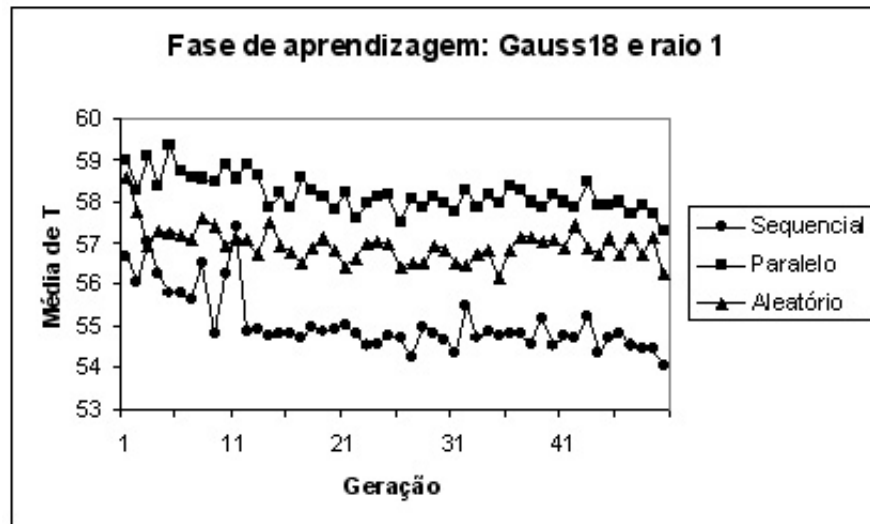


Fig. 5.8: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio 1.

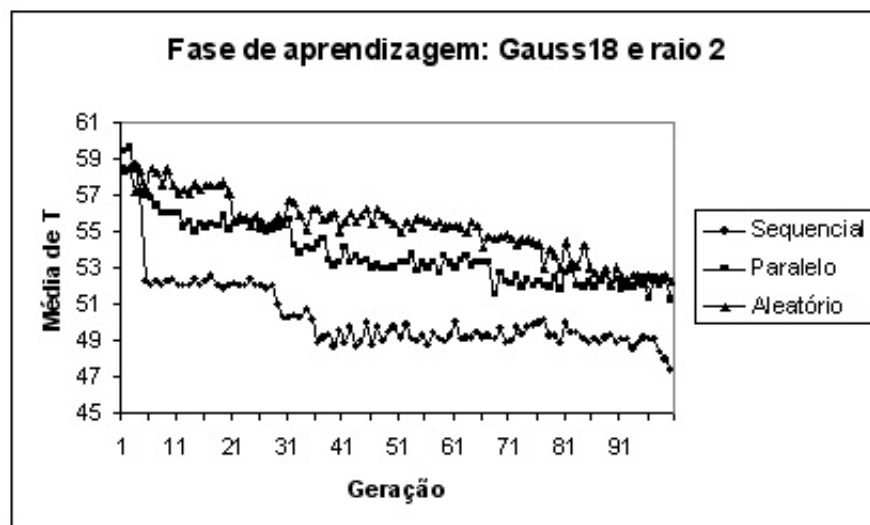


Fig. 5.9: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio $r = 2$.

de operação de AC. Como identificamos que os melhores resultados foram obtidos com o modo de operação seqüencial, optamos por não apresentar no próximo gráfico as evoluções para os modos paralelo e aleatório, para facilitar a visualização e o entendimento do leitor. Na primeira geração, o valor de aptidão do melhor indivíduo (regra) é 58,52. Nas próximas gerações, o AG modifica as regras e, na geração 460 é descoberta uma regra que converge todas as CIs para uma alocação das tarefas que representa o escalonamento ótimo ($T = 44,0$) para o Gauss18. No entanto, esta regra não passa em alguns testes em gerações subseqüentes. Apenas na geração 481 o AG descobre uma regra que

provê um escalonamento ótimo para o Gauss18 quando aplicada em todos os testes de problema (CIs) subsequentes. Para o modo de operação de paralelo, o AG descobre regras provendo soluções sub-ótimas para o problema, onde $44,0 < T \leq 45,0$. No entanto, em 1.000 gerações, nenhuma solução que leve ao escalonamento ótimo a partir de qualquer configuração inicial do AC é descoberta. E, para o modo de operação aleatório, o valor médio de T para o melhor indivíduo da última geração foi igual a 51,0.

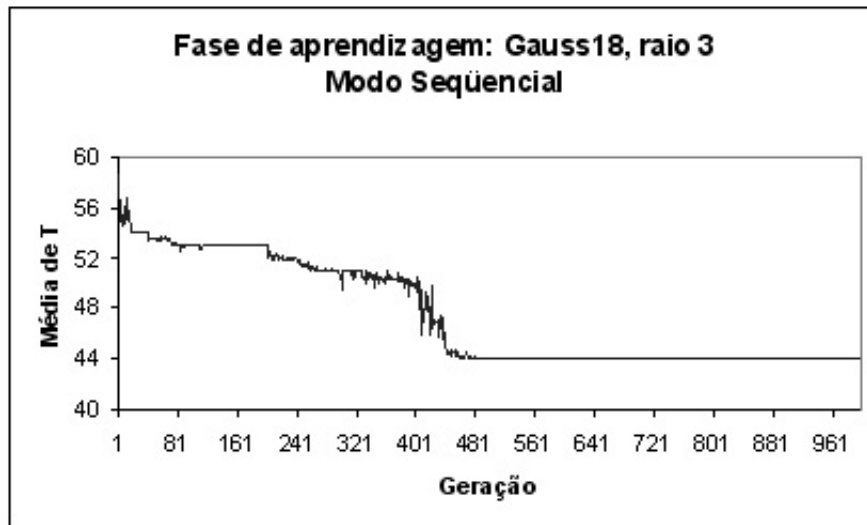


Fig. 5.10: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: raio 3 e modo de operação sequencial.

Para avaliar a qualidade das regras descobertas para o raio $r = 3$ e modo de operação sequencial, foram gerados 1.000 novos testes de problema (alocações iniciais das tarefas), sendo estes aplicados, na fase de operação, em cada uma das regras descobertas. Os resultados podem ser vistos na Figura 5.11; cerca de 106 regras foram aptas a evoluir cada alocação inicial das tarefas para um escalonamento ótimo do Gauss18.

A Figura 5.12 apresenta a evolução temporal do AC quando este é inicializado com uma das 106 regras descobertas para o Gauss18 com $T_{med} = 44,0$, e uma alocação inicial (configuração inicial) aleatória das tarefas. No passo de tempo 0, as células (reticulado) do AC estão em estados que correspondem a uma alocação das tarefas em que $T = 71,0$. O AC começa a evoluir de acordo com sua regra de transição de estados e, a partir do passo 11, é alcançada uma configuração do reticulado correspondente a uma distribuição (alocação) das tarefas entre os processadores P0 e P1 que provê o

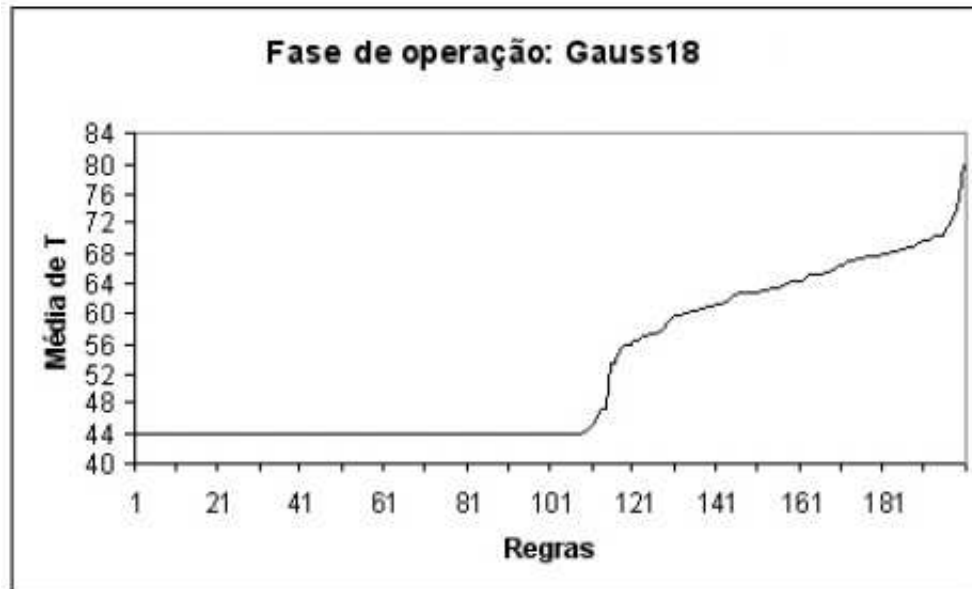


Fig. 5.11: Fase de operação do algoritmo de escalonamento para o Gauss18: modo de operação seqüencial e raio $r = 3$: média de T obtida em 1.000 CIs diferentes.

tempo total de execução $T = 44,0$.

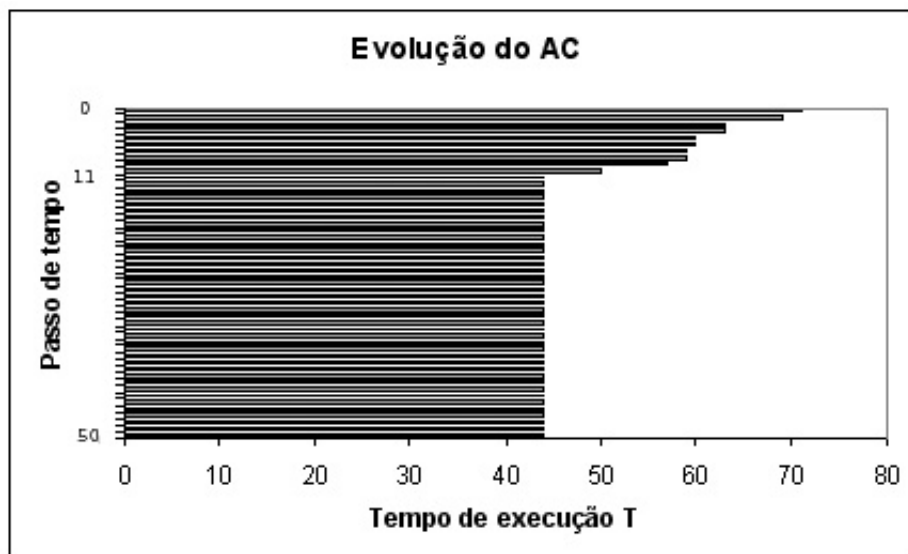


Fig. 5.12: Tempo de execução associado à evolução temporal do AC para o Gauss18.

Os resultados obtidos para o Gauss18, nos três modos de operação de AC e para os três valores de raio 1, 2 e 3, foram compatíveis com os resultados apresentados em [102].

Posteriormente, realizamos mais dois experimentos envolvendo os grafos de programa *g18* e *in-*

tree15, sendo estes experimentos semelhantes aos realizados em [103]. Nestes experimentos, o único modo de operação de AC utilizado foi o paralelo (síncrono).

Primeiramente, realizamos experimentos com o *g18*. Nestes experimentos testamos dois valores de raio r : 1 e 2. Os parâmetros utilizados para o AC foram $CIs = 50$ e $M = 50$. Para o AG, utilizamos $P = 100$, $E = 50$, $G = 100$, $p_c = 0,50$, $p_m = 0,375$ para raio 1 e $p_m = 0,09$ para raio 2.

Apenas nas regras de raio 2 foi possível convergir o reticulado do AC para uma alocação que representa o escalonamento ótimo para o *g18* ($T = 46,0$). Para $r = 1$, o tempo de execução T médio para o melhor indivíduo na última geração foi igual a 46,74. Para $r = 2$, este valor foi igual a 46,0. A Figura 5.13 apresenta a fase de aprendizagem do algoritmo de escalonamento para o *g18*, utilizando-se os valores de raio 1 e 2. Como pode ser observado para $r = 2$, o AG precisou de pouco mais de 10 gerações para descobrir uma regra de transição apta a convergir todas as CIs para uma alocação que representa o escalonamento ótimo para o *g18*.

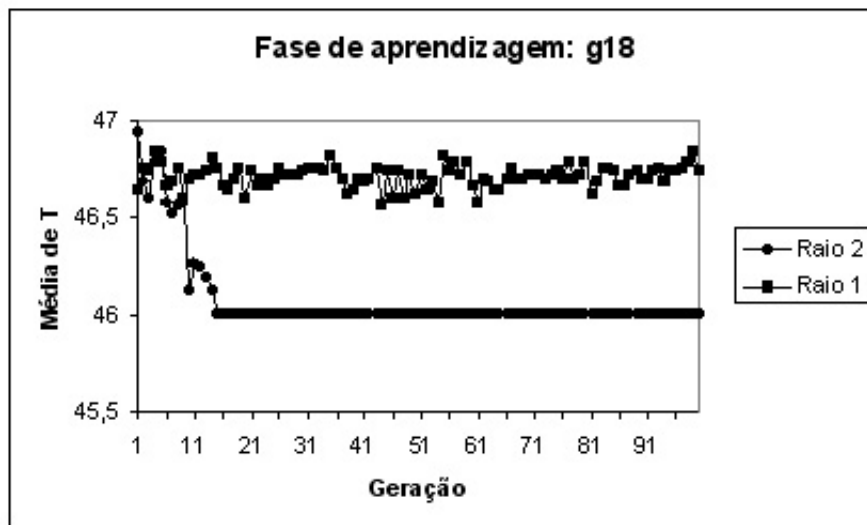


Fig. 5.13: Fase de aprendizagem do algoritmo de escalonamento para o *g18*.

Na fase de operação, para avaliar as regras descobertas para $r = 2$, foram geradas 1.000 novas alocações iniciais aleatórias (testes de problema) do *g18*. A Figura 5.14 apresenta o tempo médio de T para cada regra descoberta quando aplicada em todos os testes de problema. Como pode ser observado, cerca de 50 regras foram aptas a convergir o AC, a partir de 1.000 alocações iniciais diferentes, para uma configuração que representa o escalonamento ótimo do *g18*.

Em outro experimento, ainda referente à fase de operação, utilizamos as regras descobertas para



Fig. 5.14: Fase de operação do algoritmo de escalonamento para o *g18*.

o *g18* para resolver o problema do escalonamento para o grafo *intree15*. Sendo assim, novamente geramos 1.000 novos testes de problema (alocações iniciais do *intree15*) e os usamos para avaliar no *intree15* cada regra encontrada para o *g18*. A Figura 5.15 apresenta os resultados obtidos. O *intree15* tem $T_{ótimo} = 9,0$. Como pode ser observado, cerca de 8 regras descobertas para o *g18*, também foram aptas a encontrar o escalonamento ótimo para o *intree15*.

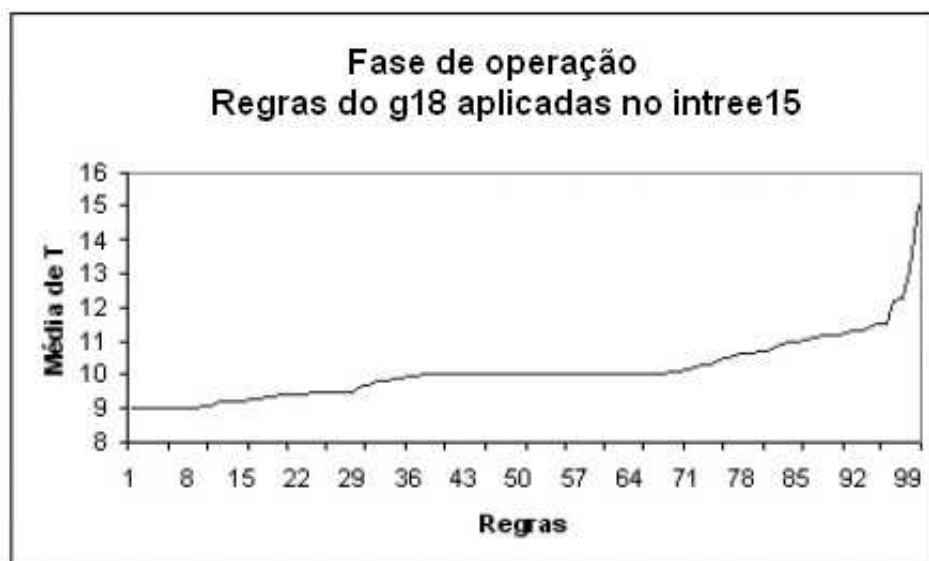


Fig. 5.15: Aplicação das regras descobertas para o *g18* na fase de operação do algoritmo de escalonamento para o *intree15*.

Os próximos experimentos foram conduzidos com os grafos de programa *tree15* e *g40* na fase de aprendizagem. Estes experimentos são similares aos apresentados em [104]. Em ambos, os parâmetros do AC foram $CIs = 50$ e $M = 50$. Para o AG foram utilizados $P = 100$, $E = 50$, $G = 100$, $p_c = 0,50$ e $p_m = 0,03$. Para o *tree15*, os experimentos mostraram que o valor de raio $r = 1$ é suficiente para descobrir regras de escalonamento. Para os três modos de operação, logo na primeira geração, o AG encontrou regras de AC aptas a encontrar o escalonamento ótimo para o *tree15* ($T = 9,0$ em um sistema com dois processadores).

O experimento seguinte foi conduzido com o grafo de programa *g40*. O valor mínimo de r que permitiu, na fase de aprendizagem, descobrir regras capazes de encontrar um escalonamento ótimo para o *g40* foi 2. A Figura 5.16 apresenta o processo evolutivo (fase de aprendizagem) para os três modos de operação de AC. Como pode ser observado, para dois modos de operação, paralelo e seqüencial, o AG descobriu regras que convergem o AC para uma configuração que corresponde ao valor ótimo para o *g40* ($T = 80,0$), e para o modo aleatório o AG descobre regras que convergem o AC para soluções apenas sub-ótimas. Na fase de operação, para avaliar a qualidade das regras descobertas para os modos paralelo e seqüencial, foram geradas 1.000 novas configurações (alocações iniciais) para o *g40*, as quais foram utilizadas para testar cada regra descoberta. Nesta fase, constatamos que o modo de operação seqüencial mostrou-se um pouco superior ao modo paralelo, onde no primeiro cerca de 63 regras foram aptas a encontrar o escalonamento ótimo para o *g40*, e no segundo cerca de 55 regras.

5.5.2 Teste da capacidade de generalização das regras descobertas através do modelo de vizinhança linear

Em todos os experimentos que simulamos, assim como nos experimentos originais descritos no Apêndice B, o AG foi apto a encontrar regras de transição capazes de resolver o problema do escalonamento para uma dada instância do problema (grafo de programa). Para avaliarmos se as regras descobertas para um dado grafo de programa podem ser aplicadas, com sucesso, para escalonar outros grafos de programa diferentes, realizamos alguns experimentos onde as melhores regras descobertas capazes de escalonar um determinado grafo de programa foram aplicadas, na fase de operação, para outros grafos de programa. Para realizar estes experimentos, utilizamos os grafos de programa Gauss18,

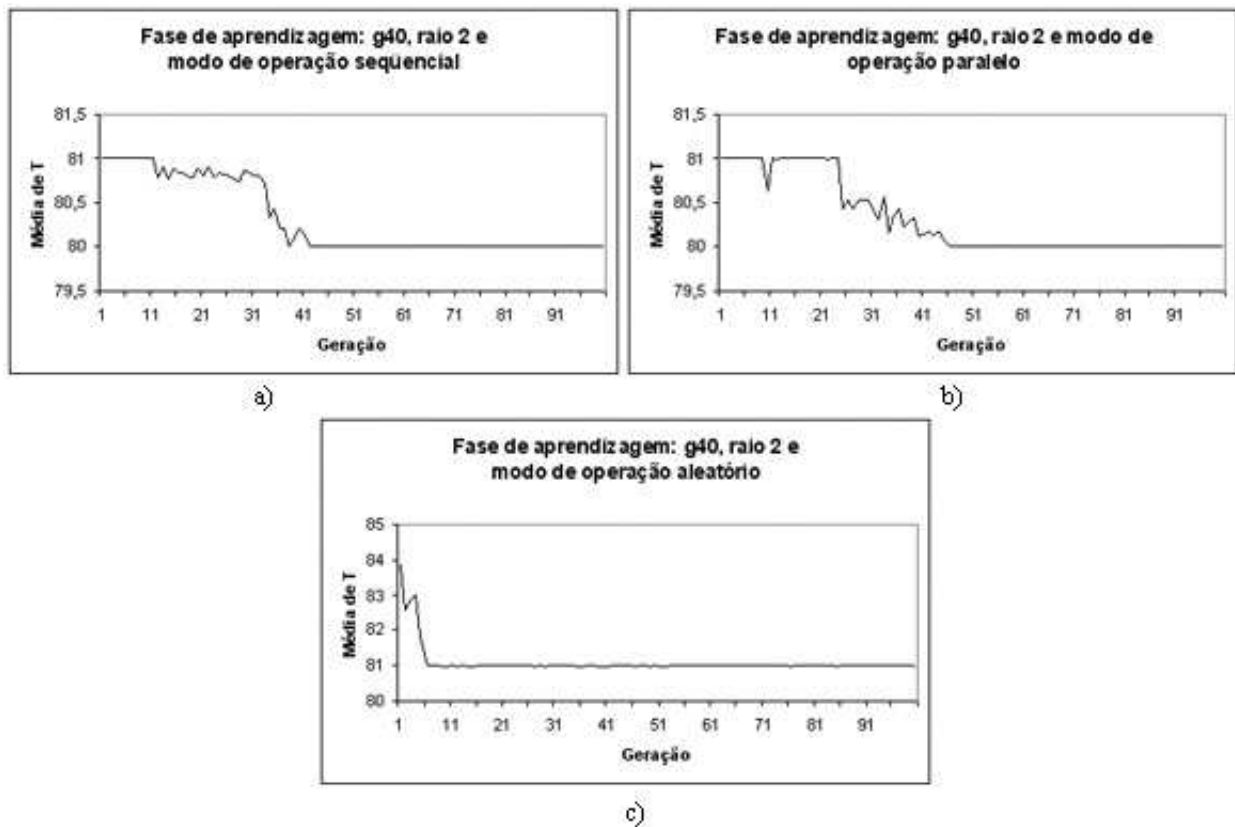


Fig. 5.16: Fase de aprendizagem do algoritmo de escalonamento para o g40: raio $r = 2$ e os três modos de operação de AC.

g18, *g40* e *tree15*.

No primeiro experimento, aplicamos as regras descobertas para o Gauss18 para os grafos *g18*, *g40* e *tree15*, onde foram gerados, na fase de operação do algoritmo de escalonamento, 1.000 testes de problema (alocações das tarefas nos processadores) para cada um destes grafos. Conforme vimos anteriormente, as melhores regras foram descobertas para o Gauss18 com a utilização do valor de raio 3 e do modo de operação sequencial. Ao aplicarmos cada uma das regras em cada teste de problema gerado para o *g18*, nenhuma delas apresentou o valor de T , na média, igual a 46,0. A melhor regra apresentou um valor médio de T (T_{med}) igual a 49,52, o valor de mínimo de T (T_{min}) igual a 46,0 e o valor máximo de T (T_{max}) igual a 56,0. O fato da regra ter encontrado o valor T_{min} igual a 46,0 significa que para algum(s) teste(s) de problema a regra foi capaz de convergir o AC para uma configuração que representa o escalonamento ótimo para o *g18*. O valor $T_{max} = 56,0$ significa que ao aplicar a regra em 1.000 testes de problema, no pior caso, a regra convergiu o AC para uma

configuração do reticulado que representa uma alocação das tarefas com $T = 56,0$. Das 200 regras de transição de AC avaliadas, cerca de 22 convergiram algum teste de problema para uma alocação das tarefas com $T_{min} = 46,0$. A Tabela 5.4 apresenta os valores de T_{med} , T_{min} , T_{max} para a melhor regra da população, e o valor $Qtde.T_{ótimo}$, que indica a quantidade de regras que conseguiram resolver pelo menos um teste de problema. Como pode ser observado, para os três grafos de programa, as regras obtidas para o Gauss18 foram capazes, em algumas configurações iniciais aleatórias (testes de problema), de convergir o AC para alocações que representam o escalonamento ótimo para cada grafo de programa. Apesar das regras não terem encontrado o escalonamento ótimo para cada grafo em todas as configurações iniciais, alguma capacidade de generalização estas regras possuem, visto que, em algum momento, o $T_{ótimo}$ foi alcançado para cada grafo.

Grafo Origem	Grafo Destino	T_{med}	T_{min}	T_{max}	$Qtde.T_{ótimo}$
Gauss18 ($T_{ótimo} = 44,0$)	g18	49,52	46,0	56,0	22
	g40	85,17	84,0	109,0	77
	tree15	9,63	9,0	13,0	139
g18 ($T_{ótimo} = 46,0$)	Gauss18	59,36	53,0	75,0	0
	g40	82,42	80,0	89,0	24
	tree15	9,0	9,0	9,0	99
g40 ($T_{ótimo} = 80,0$)	Gauss18	69,05	56,0	95,0	0
	g18	46,0	46,0	46,0	90
	tree15	9,0	9,0	9,0	100
tree15 ($T_{ótimo} = 9,0$)	Gauss18	59,26	59,0	60,0	0
	g18	46,76	46,0	53,0	36
	g40	81,0	81,0	81,0	20

Tab. 5.4: Teste da capacidade generalização das regras descobertas para os grafos de programa Gauss18, g18, g40 e tree15.

No segundo experimento, aplicamos as melhores regras descobertas para o g18 nos grafos de programa Gauss18, g40 e tree15. A Tabela 5.4 apresenta os resultados obtidos. Como pode ser visto, regras descobertas para o g18 encontraram uma solução sub-ótima para o g40 e uma solução ótima para tree15, onde em todos os testes de problema para ele criados, a regra conseguiu convergir o AC para uma alocação correspondente a um escalonamento ótimo do tree15 ($T = 9,0$). Para o Gauss18, a melhor solução encontrada não apresentou boa qualidade, visto que o valor médio de T para ela foi igual a 59,36 e, em nenhum dos 1.000 testes de problema para este grafo, a regra foi capaz de

convergir o AC para uma alocação cujo tempo total de execução T fosse igual a 44,0.

Posteriormente, foram aplicadas as 100 regras descobertas na última geração para o *g40* nos outros três grafos de programa. Conforme resultados apresentados na Tabela 5.4, foram encontradas regras capazes de resolver o problema do escalonamento tanto para o *g18* quanto para o *tree15*. Para o Gauss18 a solução encontrada não foi de boa qualidade.

Em outro experimento, aplicamos as 100 regras descobertas para o *tree15* para os demais grafos, na fase de operação. Os resultados estão listados na Tabela 5.4 e, como pode ser visto, a melhor regra para o *g18* encontrou uma solução sub-ótima com o valor médio de T igual a 46,76, e a melhor regra para o *g40* encontrou uma solução sub-ótima com o valor médio de T igual a 81,0. Para o Gauss18, como nos outros dois casos acima citados, a melhor regra encontrada para este grafo não foi de boa qualidade, sendo o valor médio de T (59,26) bem acima do valor ótimo ($T = 44,0$).

Diante dos resultados obtidos até aqui, percebemos que as regras descobertas para um grafo de programa com uma estrutura regular (*tree15*, *g18* e *g40*), apresentaram bons resultados quando aplicadas a outros grafos com estrutura também regular. Quando as regras encontradas para o Gauss18, um grafo de programa com estrutura irregular, foram aplicadas para grafos de programa com estrutura regular, resultados razoáveis foram constatados. Porém, quando regras descobertas para grafos com estrutura regular foram aplicadas para o Gauss18, os resultados obtidos não foram de boa qualidade.

Nos experimentos acima apresentados, aplicamos as regras descobertas para um determinado grafo de programa para resolver o problema do escalonamento para outros grafos de programa totalmente diferentes entre si. Porém, é interessante observar o comportamento das regras descobertas para um dado grafo de programa quando elas são aplicadas em grafos de programa semelhantes aos grafos para o qual ela foi descoberta. Assim, decidimos aplicar as regras descobertas para o Gauss18 nas quinze variações deste grafo de programa, descritas na Seção 5.3. Foram gerados 1.000 testes de problema (configurações iniciais aleatórias) para cada uma das variações do Gauss18, utilizadas para avaliar cada regra descoberta. Os resultados obtidos estão apresentados na Tabela 5.5. Como pode ser visto, foram identificadas regras capazes de encontrar o escalonamento ideal para os grafos de programa Gauss18_7 e o Gauss18_10. Chamamos de escalonamento ideal, aquele que leva o AC a convergir para um tempo igual ao obtido pelo AG_{ref} , que não podemos afirmar que é o ótimo. Assim, os valores obtidos pelo AG de referência apresentados na Tabela 5.3, foram reproduzidos na Tabela

5.5 para facilitar a compreensão. Para os grafos de programa Gauss18_2, Gauss18_3, Gauss18_4, Gauss18_5 e Gauss18_11 todas as regras encontraram tempos bem acima do ideal e, para os demais grafos, soluções razoáveis foram encontradas (próximas do valor ideal).

Grafo	T_{AG_Ref}	T_{med}	T_{min}	T_{max}	$Qtde.T_{ótimo}$
Gauss18_1	47,0	48,0	48,0	48,0	0
Gauss18_2	47,0	52,0	52,0	52,0	0
Gauss18_3	47,0	55,0	55,0	55,0	7
Gauss18_4	47,0	55,0	55,0	55,0	0
Gauss18_5	50,0	55,0	55,0	55,0	0
Gauss18_6	47,0	48,0	48,0	48,0	0
Gauss18_7	44,0	44,0	44,0	44,0	177
Gauss18_8	44,0	47,0	47,0	47,0	0
Gauss18_9	47,0	48,0	48,0	48,0	7
Gauss18_10	44,0	44,0	44,0	44,0	174
Gauss18_11	47,0	51,0	51,0	51,0	10
Gauss18_12	47,0	48,0	48,0	48,0	0
Gauss18_13	44,0	47,0	47,0	47,0	0
Gauss18_14	46,0	48,0	48,0	48,0	0
Gauss18_15	47,0	48,0	48,0	48,0	7
Média	46,33	49,20	49,20	49,20	-

Tab. 5.5: Valores obtidos através da aplicação, na fase de operação, das regras descobertas para o Gauss18 nas quinze variações do mesmo.

Portanto, analisando-se todos os resultados acima apresentados, nós entendemos que alguma capacidade de generalização as regras de transição de AC descobertas possuem, visto que algumas delas conseguem resolver o problema do escalonamento para diferentes grafos de programa. Os resultados obtidos são promissores e abrem espaço para que, por exemplo, novas abordagens sejam criadas visando melhorar tal capacidade de generalização das regras de transição de AC.

No modelo de vizinhança linear de AC, a estrutura não-linear de um grafo de programa é aproximada para uma estrutura linear de AC unidimensional. Desta forma, a vizinhança de uma determinada tarefa do grafo de programa não é formada por tarefas que realmente estão próximas da tarefa em questão mas sim, pela proximidade das células no reticulado do AC. Por exemplo, considerando-se um valor de raio $r = 1$, a vizinhança da tarefa 3 será sempre formada pelas tarefas 2, 3 e 4, mesmo que as tarefas 2 e 4 não tenham nenhuma ligação com a tarefa 3. Além disso, uma alteração da numeração das tarefas no grafo de programa não altera a vizinhança das tarefas/células no reticulado.

Logo, acreditamos que os modelos de vizinhança não-lineares, que consideram a estrutura do grafo de programa e a proximidade das tarefas neste grafo, são mais adequados para a resolução do problema.

A seguir, são apresentados resultados obtidos através da utilização dos modelos de vizinhança não-lineares e o modo de operação seqüencial de AC.

5.6 Resultados com os modelos de vizinhança não-linear e modo de operação seqüencial

Nesta seção, os resultados experimentais apresentados foram obtidos utilizando-se, para o AC, os modelos de vizinhança não-lineares e o modo de operação seqüencial.

Nos experimentos publicados [94, 95, 96, 99] e também nos que simulamos como parte de nosso trabalho inicial, percebemos que o modo de operação de AC que apresentou os melhores resultados foi o seqüencial. Por isso, optamos por utilizar este modo de operação nos primeiros experimentos com os modelos não-lineares. Posteriormente, retomamos os estudos com o modo de operação paralelo, o qual julgamos também ser fundamental para o algoritmo de escalonamento baseado em ACs, uma vez que o paralelismo, característica fundamental dos ACs, pode contribuir para uma melhor performance do algoritmo. Os experimentos com o modo paralelo serão apresentados no Capítulo 6.

5.6.1 Experimentos reproduzidos e novos

Nesta seção, apresentamos alguns experimentos que realizamos, baseados nos experimentos apresentados em [94, 95, 96] e descritos no Apêndice B. Nestes experimentos, os principais grafos de programa utilizados foram o Gauss18, *g18*, *g40* e *tree15*.

Em todos os experimentos, os parâmetros utilizados para o AC foram $CI_s = 25$ e $M = 50$. Para o AG, que utiliza os mesmos passos descritos na Seção 5.1.4, utilizamos $P = 100$, $E = 10$, $p_c = 0,90$, $G = 100$, $p_m = 0,012$ para regras formadas com a *Vizinhança Seleccionada* e $p_m = 0,024$ para regras formadas com a *Vizinhança Totalística*. Com exceção do experimento realizado para o grafo de programa *g40*, que utilizou $P = G = 200$, todos os outros experimentos utilizaram os parâmetros do AG acima descritos.

No primeiro experimento realizado, foi utilizado o grafo de programa Gauss18 e o modelo de vizinhança selecionada de AC. Na formação da vizinhança de uma dada tarefa k , foram considerados os atributos co-nível estático, custo computacional e custo de comunicação para os conjuntos $predecessoras(k)$, $irmãs(k)$ e $sucessoras(k)$, respectivamente. A política de escalonamento utilizada foi “a tarefa com maior nível dinâmico é executada primeiro”. Essa política foi utilizada em todos os experimentos com o Gauss18. A Figura 5.17a apresenta a fase de aprendizagem do algoritmo de escalonamento para este experimento. Na primeira geração, o valor médio de T para o melhor indivíduo (regra) da população é 58,76. Nas próximas gerações, o AG modifica as regras e, na geração 14, é descoberta uma regra de transição de AC capaz de evoluir, por 50 passos de tempo (M), cada uma das 25 CIs para configurações que representam um escalonamento ótimo para o Gauss18 ($T = 44,0$).

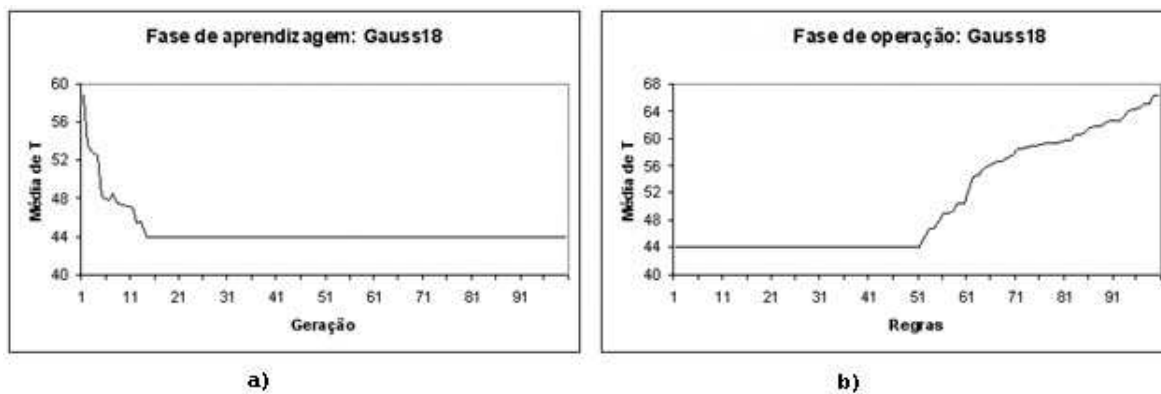


Fig. 5.17: Fase de aprendizagem e fase de operação do algoritmo de escalonamento para o Gauss18: Vizinhança Selecionada, modo de operação sequencial e três atributos diferentes para a formação da vizinhança.

A qualidade das regras é avaliada na fase de operação do algoritmo de escalonamento. Então, nesta fase, geramos 100 novos testes de problema, e os usamos para testar cada regra descoberta. A Figura 5.17b apresenta os resultados obtidos (média de T) para cada regra da população: cerca de 50 regras foram capazes de encontrar um escalonamento ótimo para cada teste de problema. Ou seja, quando aplicada em cada CI, a regra foi capaz de evoluir, por 50 passos de tempo, para uma configuração final em que $T = 44,0$. A Figura 5.18 apresenta a evolução do AC utilizando uma das melhores regras da população e a configuração inicial aleatória $CI = 100111000010011011$. No passo 0, o valor de $T = 75,0$. O AC evolui, de acordo com a regra de transição, alterando a sua configuração e, conseqüentemente o valor de T , até que no passo 14 é alcançada a configuração

101110101101011111, que permanece até o passo 50, e que representa um escalonamento ótimo para o Gauss18.

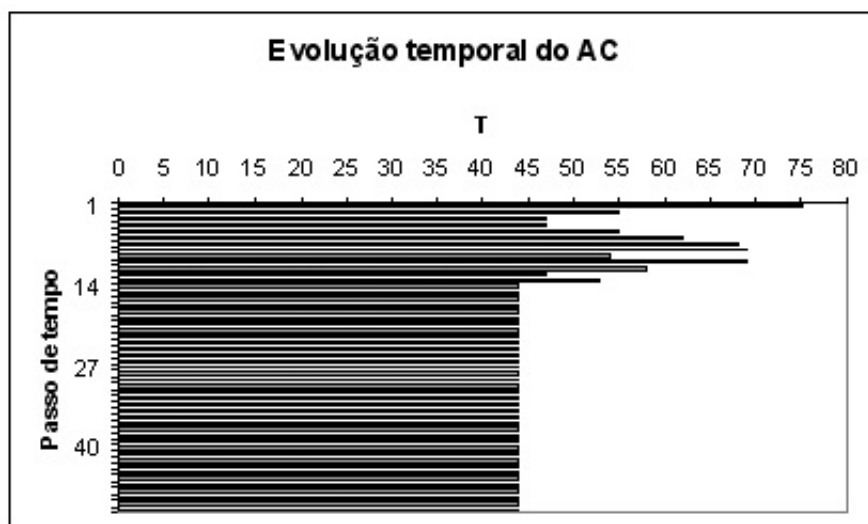


Fig. 5.18: Evolução temporal do AC com uma regra que escala o Gauss18.

Realizamos outro experimento semelhante ao descrito acima, porém, a diferença é que utilizamos a *Vizinhança Totalística*. A Figura 5.19 apresenta a fase de aprendizagem correspondente a este experimento. Na primeira geração, o valor médio de T para o melhor indivíduo é 55,76. O AG modifica as regras e, a partir da geração 44, a média de T para o melhor indivíduo oscila entre 47,0 e 49,0, descobrindo uma regra na última geração que leva ao valor médio de $T = 47,36$. Assim, utilizando-se este modelo de vizinhança, não foi possível encontrar uma regra de transição de AC capaz de escalonar, com sucesso, o Gauss18 nas 25 CIs utilizadas como teste. Por isso, optamos por não realizar a segunda etapa do algoritmo, onde as regras são avaliadas na fase de operação.

No próximo experimento utilizamos a *Vizinhança Seleccionada*, porém, ao invés de serem utilizados três atributos diferentes para a formação da vizinhança, utilizamos um único atributo para os conjuntos que formam a vizinhança de cada tarefa: o nível dinâmico. Optamos por esse atributo, por ser o mesmo utilizado na política de escalonamento. Não encontramos, na literatura, um experimento similar a este. O nível dinâmico é calculado em função da alocação das tarefas nos processadores. Trocando-se uma tarefa de processador, o nível dinâmico deve ser calculado novamente. Durante a evolução temporal do AC, as tarefas podem ser alocadas em diferentes processadores, fazendo com que o nível dinâmico seja recalculado a cada nova alocação. Dessa forma, o tempo de processamento

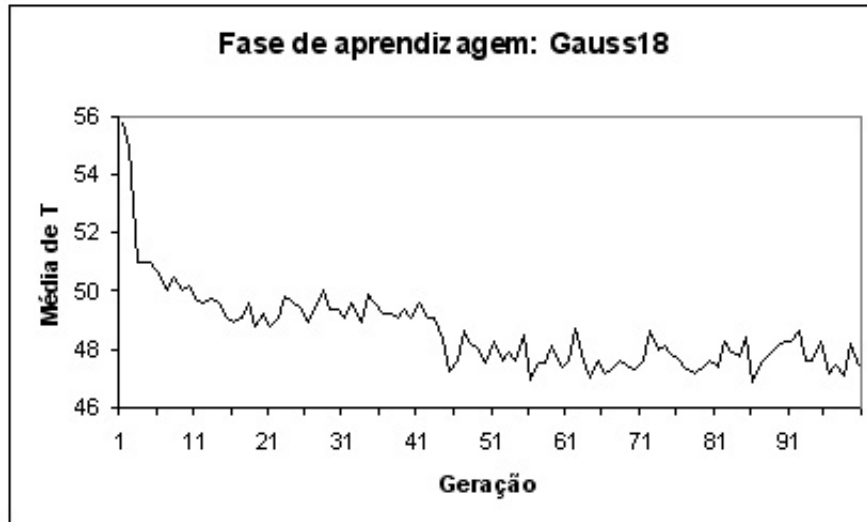


Fig. 5.19: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.

do algoritmo é maior quando comparado a experimentos que utilizam atributos estáticos na formação da vizinhança, sendo este o provável motivo da não utilização deste atributo em experimentos publicados. A Figura 5.20a apresenta a fase de aprendizagem para este experimento. A média de T para o melhor indivíduo na primeira geração é 56,84. Durante as próximas gerações o AG modifica as regras e a média de T para o melhor indivíduo diminui. Há algumas oscilações e na geração 20 o AG descobre uma regra que converge o reticulado do AC para uma alocação ótima com $T = 44,0$. Isto significa que a regra leva o reticulado do AC a convergir para uma configuração que representa uma alocação ótima do Gauss18, a partir de todas as 25 configurações iniciais usada para avaliar a regra. Esta regra, entretanto, não passa em um teste criado na geração 70. O AG rapidamente modifica a regra, provendo uma nova regra que passa em todos os testes subseqüentes.

A Figura 5.20b apresenta a fase de operação do algoritmo para este experimento. Ao aplicarmos as regras descobertas em 100 novas configurações iniciais (testes de problema) para o Gauss18, cerca de 28 regras foram capazes de convergir o reticulado do AC para configurações que representam o escalonamento ótimo.

Um último experimento foi realizado para o Gauss18 e a *Vizinhança Totalística*, utilizando apenas o atributo nível dinâmico para formação da vizinhança de uma dada tarefa k do grafo de programa. A Figura 5.21 apresenta o T médio para o melhor indivíduo (regra) da população durante o processo

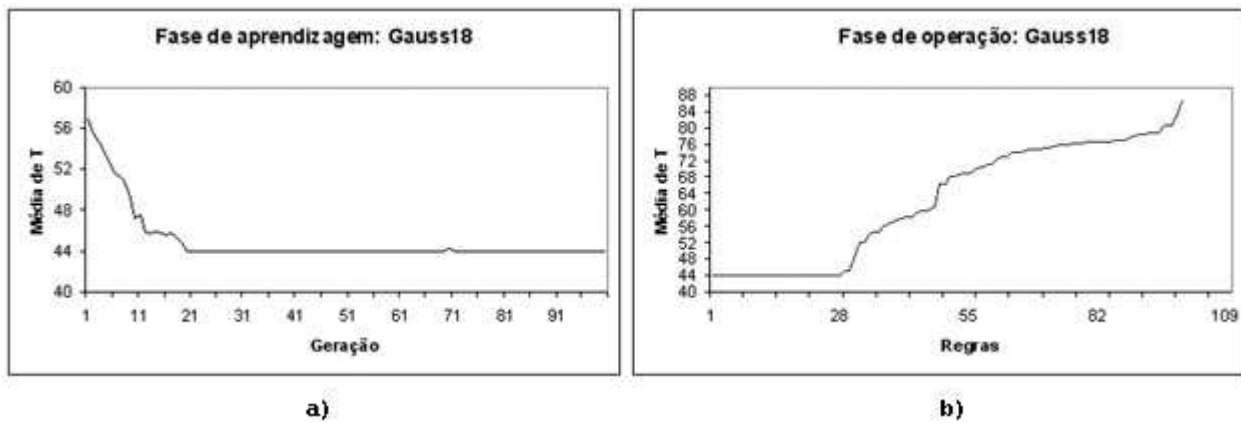


Fig. 5.20: Fase de aprendizagem e fase de operação do algoritmo de escalonamento para o Gauss18: Vizinhaça Seleccionada, modo de operação seqüencial e atributo nível dinâmico para a formação da vizinhaça.

evolutivo. Na primeira geração, a melhor regra da população possui o valor médio de aptidão igual a 54,40. O AG, durante as próximas gerações, modifica as regras e, a partir da geração 26, a melhor regra possui $44,0 \leq T \leq 45,0$ até que, na última geração, o valor médio de T para a melhor regra é 44,36. Assim como no experimento anterior com o modelo de vizinhaça totalística, não foi descoberta nenhuma regra capaz de convergir todas as configurações iniciais do reticulado do AC para configurações finais que representam o escalonamento ótimo para o Gauss18. Dessa forma, também não foi realizado nenhum experimento na fase de operação.

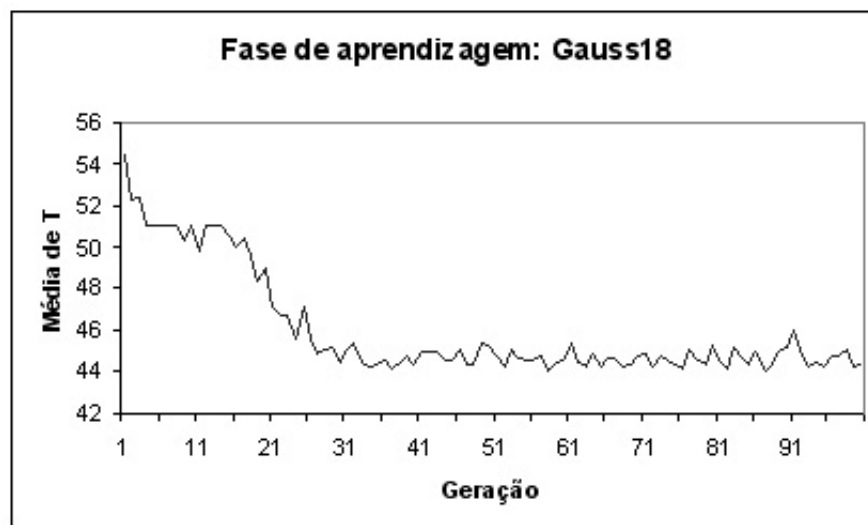


Fig. 5.21: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhaça Totalística, modo de operação seqüencial e atributo nível dinâmico para a formação da vizinhaça.

O experimento seguinte foi realizado com o grafo de programa *g40*. A vizinhança do AC utilizada foi a *Vizinhança Seleccionada*, onde os atributos utilizados na formação da vizinhança foram o nível estático, custo computacional e custo de comunicação para os conjuntos de *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)*, respectivamente. A política de escalonamento utilizada foi “a tarefa com maior nível dinâmico é executada primeiro”. A Figura 5.22a mostra que regras de transição de AC provendo um escalonamento ótimo com o tempo total de execução $T = 80,0$ foram descobertas após 19 gerações. A qualidade das regras descobertas, verificada na fase de operação, é apresentada na Figura 5.22b. Foram geradas 100 novas configurações iniciais (testes de problema) para avaliar cada regra descoberta na fase de aprendizagem. Podemos observar que cerca de 36 regras foram capazes de evoluir, por 50 passos de tempo, todas as 100 CIs para configurações finais que representam alocações ótimas para o *g40*.

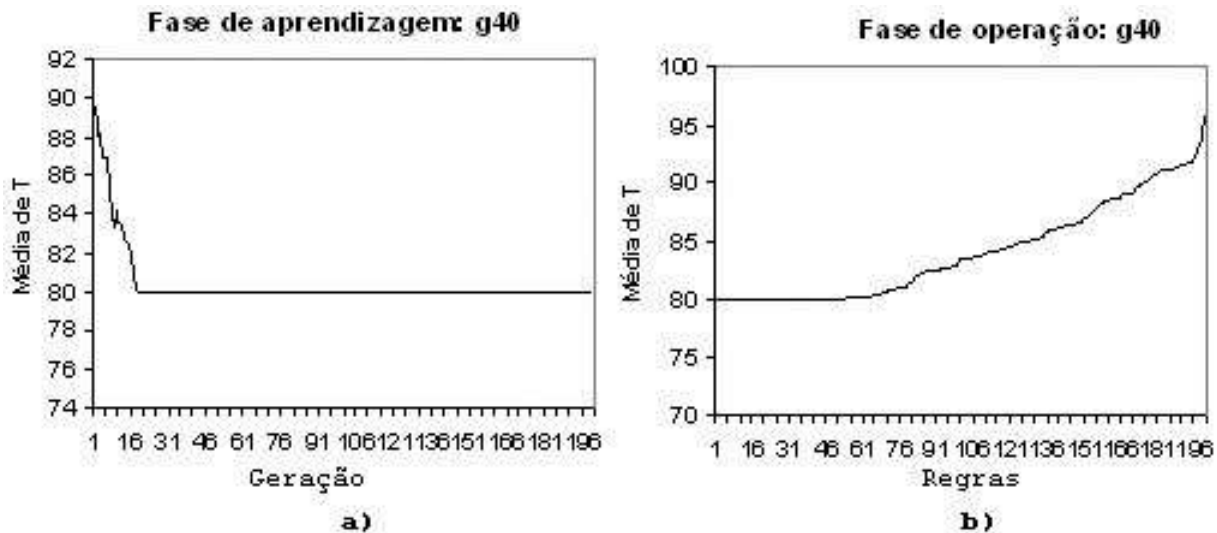


Fig. 5.22: Fase de aprendizagem e de operação do algoritmo de escalonamento para o *g40*: Vizinhança Seleccionada, modo de operação sequencial e três atributos diferentes para a formação da vizinhança.

Para o experimento com o grafo de programa *g18*, utilizamos a *Vizinhança Totalística* e a política de escalonamento “a tarefa com o maior nível estático é executada primeiro”. Os mesmos atributos utilizados para a formação da vizinhança no experimento com o *g40* foram utilizados neste experimento. Observando a Figura 5.23a, percebe-se que o AG necessita de menos de 5 gerações para descobrir, na fase de aprendizagem, uma regra de transição de AC provendo uma solução ótima com $T = 46,0$. Avaliando a qualidade das regras em 100 novas configurações iniciais, na fase de opera-

ção, podemos observar (Figura 5.23b) que cerca de 54 regras encontradas na fase de aprendizagem encontram soluções ótimas na fase de operação.

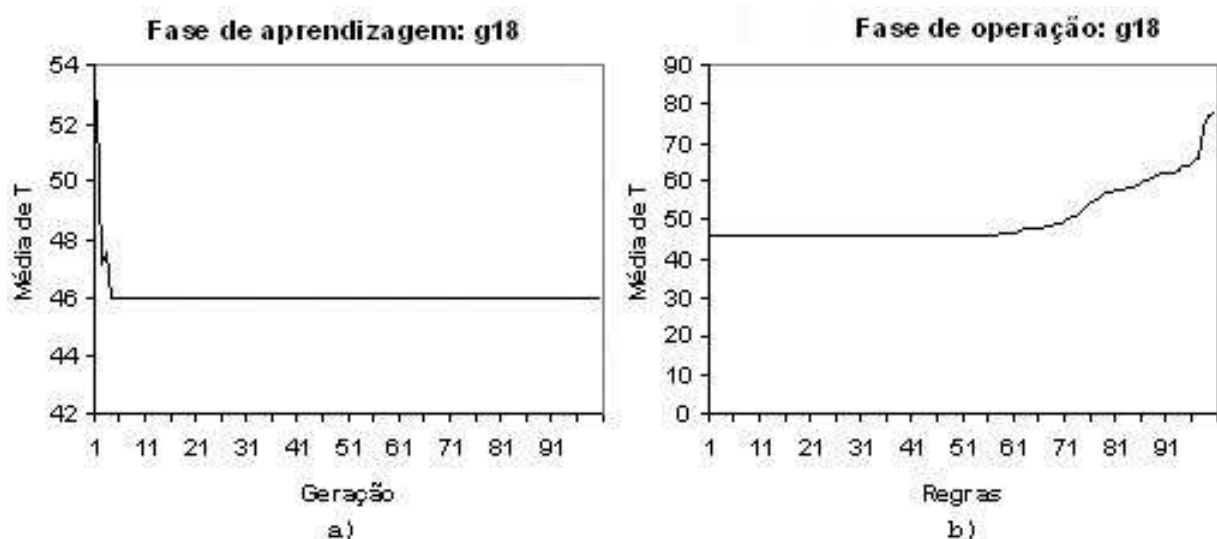


Fig. 5.23: Fases do algoritmo de escalonamento para o *g18*: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.

O experimento realizado com o grafo de programa *tree15* foi conduzido com a vizinhança de AC totalística e a política de escalonamento “a tarefa com menor número de ordem é executada primeiro”. Foram utilizados também os atributos co-nível estático, custo computacional e custo de comunicação para a formação da vizinhança. O AG não precisou mais que 4 gerações para descobrir uma regra de transição de AC provendo um escalonamento ótimo para o *tree15*, como mostra a Figura 5.24a. Na fase de operação (Figura 5.24b), foram geradas 100 novas configurações iniciais para avaliar cada regra descoberta na fase anterior. Das 100 regras na população, cerca de 43 foram capazes de forçar cada uma das 100 novas CIs a convergir para configurações que representam o escalonamento ótimo para o *tree15*.

5.6.2 Teste da capacidade de generalização das regras descobertas através do modelos de vizinhança não-lineares

Na Seção 5.5.1, apresentamos alguns experimentos realizados para avaliar a capacidade de generalização das regras de transição de AC, quando aplicadas, na fase de operação, para diferentes grafos de

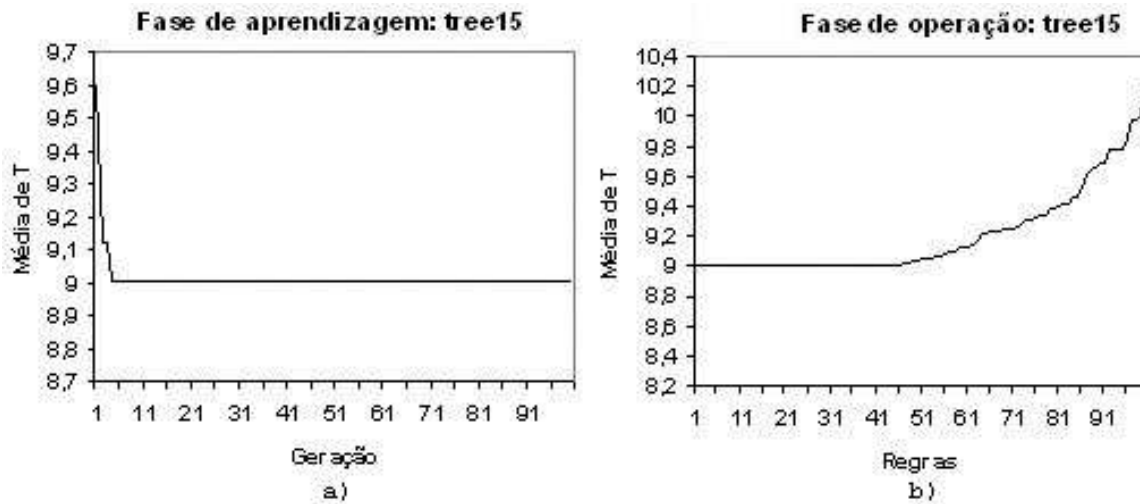


Fig. 5.24: Fases do algoritmo de escalonamento para o *tree15*: Vizinhança Totalística, modo de operação seqüencial e três atributos diferentes para a formação da vizinhança.

programa. Nestes experimentos, o modelo de vizinhança utilizado foi o linear. Nesta seção, também realizamos experimentos para verificar a capacidade de generalização das regras de transição, porém, utilizamos aqui os modelos de vizinhança não-lineares.

Os principais grafos de programa utilizados foram o Gauss18, o *g18*, *g40* e *tree15*. Os resultados de todos os experimentos estão listados na Tabela 5.6. A coluna *Grafo Origem* apresenta qual grafo de programa foi utilizado na fase de aprendizagem, a coluna *Modelo Vizinhança* identifica qual o modelo de vizinhança foi utilizado e a coluna *Atributo(s) Vizinhança* identifica se foi utilizado apenas o atributo nível dinâmico ou os três atributos diferentes (co-nível estático, custo computacional e custo de comunicação) para a formação da vizinhança. Para avaliar a capacidade de generalização, as regras descobertas para o *Grafo Origem* foram utilizadas no escalonamento de cada *Grafo Destino*. Para essa avaliação, na fase de operação, foram gerados 100 testes de problema para cada *Grafo Destino*, representando alocações iniciais aleatórias dos mesmos, nos processadores P0 e P1.

A Tabela 5.6 apresenta os melhores resultados obtidos para cada grafo de programa, isto é, apresenta para a melhor regra da população, o valor médio de T (T_{med}) obtido nos 100 testes de problema, o menor valor de T obtido (T_{min}), o maior valor de T obtido (T_{max}) e a quantidade de regras da população que conseguiram resolver pelo menos um teste de problema para o referido grafo ($Qtde.T_{ótimo}$). Conforme pode ser observado, resultados razoáveis foram obtidos somente quando as regras descobertas para o Gauss18, *g18* e *g40* foram aplicadas no escalonamento do *tree15*. Nos demais casos,

os resultados obtidos não foram satisfatórios, onde o valor médio de T (T_{med}) para cada grafo de programa foi significativamente superior quando comparado aos valores de referência apresentados na Tabela 5.3.

Grafo Origem	Modelo Vizinhança	Atributo(s) Vizinhança	Grafo Destino	T_{med}	T_{min}	T_{max}	$Qtd.T_{ótimo}$
Gauss18	Selecionada	3 atributos	g18	58.46	52.0	63.0	0
			g40	97.51	80.0	128.0	19
			tree15	9.84	9.0	12.0	100
Gauss18	Selecionada	Nível Dinâmico	g18	59.81	49.0	88.0	1
			g40	95.37	85.0	128.0	0
			tree15	9.91	9.0	12.0	98
Gauss18	Totalística	3 atributos	g18	56.25	55.0	70.0	1
			g40	90.65	82.0	109.0	16
			tree15	10.02	9.0	12.0	93
Gauss18	Totalística	Nível Dinâmico	g18	57.84	46.0	87.0	28
			g40	93.82	84.0	121.0	6
			tree15	9.90	9.0	11.0	95
g18	Totalística	3 atributos	Gauss18	72.78	59.0	93.0	0
			g40	87.41	81.0	100.0	0
			tree15	9.63	9.0	12.0	98
tree15	Totalística	3 atributos	Gauss18	62.85	56.0	76.0	0
			g18	59.08	51.0	66.0	0
			g40	105.20	88.0	125.0	0
g40	Selecionada	3 atributos	Gauss18	58.36	53.0	65.0	0
			g18	61.27	56.0	82.0	0
			tree15	9.39	9.0	15.0	151

Tab. 5.6: Teste da capacidade de generalização das regras de transição.

Diante dos resultados experimentais publicados, apresentados no Apêndice B, e dos resultados experimentais por nós obtidos (Seções 5.5.1 e 5.6.1), percebemos que o grafo de programa que representa uma maior dificuldade para a busca de regras de transição de AC aptas a escalonar é o Gauss18, sendo que, possivelmente, essa dificuldade esteja relacionada à estrutura irregular deste grafo de programa [104]. Também pudemos perceber que os melhores resultados para o Gauss18 foram obtidos com a utilização do modo de operação seqüencial do AC. Por outro lado, o grafo de programa que mostrou-se mais fácil de ser escalonado foi o *tree15*, visto que para todos os modelos de vizinhança e atributos, bons resultados foram obtidos. Acreditamos que esta facilidade esteja relacionada à estrutura regular desse grafo.

A partir dessa análise, realizamos experimentos, que serão apresentados no Capítulo 6, visando melhorar a capacidade de generalização das regras de transição de ACs. Nesses experimentos, nos concentramos no grafo de programa Gauss18 e suas variações, e no modo de operação (atualização) seqüencial de AC, em virtude dos melhores resultados para o Gauss18 terem sido alcançados, até então, com a utilização deste modo de operação.

Duas novas abordagens foram avaliadas para a fase de aprendizagem das regras: a Evolução Conjunta e a Coevolução.

Capítulo 6

Novas abordagens propostas para a melhoria da capacidade de generalização de regras

A capacidade de generalização das regras dos ACs evoluídas na fase de aprendizagem é crucial para o algoritmo de escalonamento discutido no capítulo anterior. Se as regras descobertas pelo algoritmo genético não puderem ser utilizadas em diferentes grafos de programas, todo o esforço computacional empregado na busca por elas não é justificável.

Neste capítulo apresentamos duas novas abordagens que propusemos para o escalonador de tarefas baseado em ACs. A utilização destas duas abordagens na fase de aprendizagem de regras contribuiu para uma melhoria na capacidade de generalização das regras dos ACs, o que representa a principal contribuição do nosso trabalho.

Além dessas duas novas abordagens, apresentamos experimentos que realizamos com o modo de operação paralelo (síncrono) de ACs. Poucos experimentos com este modo de operação foram publicados até o momento e, entendemos que o mesmo é importante para o escalonador baseado em ACs, uma vez que o paralelismo é característica fundamental para a implementação prática dos ACs, contribuindo inclusive para uma melhor performance do algoritmo.

Este capítulo está organizado da seguinte maneira: a Seção 6.1 apresenta a nova abordagem que chamamos de *Evolução Conjunta*, onde as regras de transição são evoluídas, na fase de aprendizagem do escalonador, utilizando-se mais de um grafo de programa. A Seção 6.2 apresenta a outra nova abordagem, que faz uso do Algoritmo Genético Coevolutivo (AGC), proposta também para a fase de

aprendizagem do escalonador, em substituição ao AG padrão utilizado em [94]. Nas Seções 6.1 e 6.2, o modo de operação de ACs utilizado foi o sequencial. A Seção 6.3 apresenta resultados obtidos com o modo de operação paralelo, inclusive utilizando as novas abordagens apresentadas nas Seções 6.1 e 6.2.

6.1 Evolução Conjunta: uma nova abordagem de avaliação para a fase de aprendizagem

Em todos os trabalhos anteriores [94, 95, 96, 99], um único grafo de programa é utilizado para descobrir regras de ACs na fase de aprendizagem de regras. Ou seja, o AG evolui as regras calculando-se a aptidão das mesmas baseando-se em um único grafo. Nós chamamos esta abordagem de *Evolução Simples*. Resultados publicados e os experimentos que relatamos no Capítulo 5 mostraram que o escalonador baseado em ACs foi apto a descobrir regras de transição que podem, com sucesso, escalonar diversos grafos de programa estudados anteriormente na literatura. Porém, quando as regras evoluídas descobertas para um grafo de programa específico foram utilizadas na tentativa de resolver o problema do escalonamento para outros grafos de programa, resultados conclusivos sobre a capacidade de generalização das regras não foram publicados até o momento. Os trabalhos publicados são promissores, mas ainda não apresentaram a performance desejada.

Uma observação que chamou a nossa atenção foi o fato das regras evoluídas para o Gauss18 não retornarem bons resultados mesmo quando aplicadas a outros grafos de programa muito similares a ele, conforme apresentado na Seção 5.5.2 para o modelo de vizinhança linear. Os resultados obtidos com a aplicação das regras evoluídas para os modelos não-lineares de vizinhança não foram apresentados no Capítulo 5. Os mesmos serão apresentados e discutidos de forma comparativa nas próximas sub-seções. Consideremos como exemplo o Gauss18_8, que é uma pequena variação do Gauss18 com $T_{AG} = 44,0$, conforme a Tabela 5.3. Ao utilizarmos as 100 regras descobertas na fase de aprendizagem para o Gauss18, com o modelo de Vizinhança Seleccionada e três atributos, apresentado na Seção 5.6.1, no escalonamento do Gauss18_8, o melhor resultado encontrado foi $T = 53,0$ e o pior foi $T = 62,0$.

A capacidade de generalização é uma característica muito importante das regras evoluídas na fase

de aprendizagem. O alto esforço computacional necessário na descoberta das regras de transição é justificado somente se estas regras forem aptas a serem utilizadas em novos problemas. Caso contrário, um AG pode ser diretamente utilizado na busca de configurações ótimas de cada grafo de programa independentemente, sem a necessidade de envolver os ACs no modelo. Um exemplo dessa abordagem direta pelo AG foi apresentado na Seção 4.3.1. O propósito de se utilizar regras de transição de ACs é a possibilidade de reutilização destas regras em novos problemas, sem a necessidade de um novo processo de aprendizagem evolutivo, o que demandaria um grande esforço computacional.

Uma importante observação que fizemos sobre os estudos relatados em [96, 104] é que pouca informação sobre esta capacidade de generalização foi apresentada, e poucos exemplos de reutilização das regras foram de fato verificados. No capítulo anterior, apresentamos alguns experimentos aplicando as regras evoluídas para o Gauss18, na fase de aprendizagem, para grafos de programa totalmente diferentes deste grafo. Concluímos que resultados razoáveis (valores sub-ótimos de T) foram obtidos somente em grafos de programa onde uma solução ótima era fácil de alcançar, por exemplo, o *tree15*. Então, nos concentramos na seguinte questão: se é desejado que uma regra de transição de AC evoluída com base em um grafo de programa específico tenha alguma capacidade de generalização, esta regra deve retornar, no mínimo, resultados razoáveis quando aplicada em novos grafos de programa que são variações do grafo de programa original.

Propusemos uma nova abordagem para a fase de aprendizagem do algoritmo de escalonamento baseado em AC, que chamamos de *Evolução Conjunta*. Nesta abordagem, o valor da aptidão para uma regra não deve ser baseado apenas em um único grafo de programa, mas também em algumas de suas variações. Nos testes que realizamos para avaliar a nova abordagem, consideramos o Gauss18 como o principal grafo de programa e mais outras cinco variações foram utilizadas na evolução. As regras são avaliadas com base nos seis grafos de programa. O valor da aptidão para uma regra é o somatório dos valores de T correspondente para cada grafo de programa. As regras com o menor valor de avaliação são consideradas melhores, portanto, mais gerais.

Na fase de operação, as regras descobertas na fase anterior são aplicadas em novas alocações iniciais aleatórias para cada grafo de programa que participou da fase de aprendizagem e também são aplicadas em outros grafos de programa, que são outras variações do original, para testar a qualidade das regras descobertas. Esperamos que as regras evoluídas conjuntamente sejam mais gerais que as

regras evoluídas com base em um único grafo.

Realizamos experimentos com a evolução conjunta para os dois modelos de vizinhança não-linear, Totalística e Seleccionada, nos quais o principal grafo de programa adotado foi o Gauss18. Os outros grafos utilizados nos experimentos foram as variações do Gauss18, descritas na Seção 5.3.

Em todos os experimentos com a Evolução Conjunta apresentados nessa dissertação, as regras de AC foram evoluídas calculando-se seus valores de aptidão baseado não somente no Gauss18, mas também nos grafos de programa Gauss18_1, Gauss18_2, Gauss18_3, Gauss18_4 e Gauss18_5. Estas cinco variações foram escolhidas entre as variações analisadas devido a elas serem as que mais se diferenciam (maior número de modificações) do Gauss18 original. Baseado na nova abordagem, o valor de aptidão para cada regra é o somatório de T para cada grafo de programa.

Os parâmetros para o AG e AC utilizados foram $P = 100$, $E = 10$, $p_c = 0,90$, $p_m = 0,012$, $G = 100$, $CIs = 25$ e $M = 50$. A política de escalonamento utilizada foi “a tarefa com o maior nível dinâmico é executada primeiro”. A seguir, apresentamos os resultados de quatro experimentos realizados com a Evolução Conjunta.

6.1.1 Vizinhança selecionada e atributo nível dinâmico

Na Seção 5.6.1, apresentamos um experimento onde foi utilizado o grafo de programa Gauss18, o modelo de vizinhança selecionada e um único atributo utilizado para a formação da vizinhança: o nível dinâmico. Identificamos que cerca de 28 das 100 regras descobertas foram aptas a encontrar um escalonamento ótimo ($T = 44,0$) para o Gauss18 em todos os 100 testes de problema (CIs) avaliados. Outros testes foram realizados, onde as regras evoluídas foram aplicadas nas quinze variações do Gauss18, descritas na Tabela 5.2. Os resultados estão apresentados na Tabela 6.1 (Evolução Simples). Com o objetivo de avaliar a qualidade destas regras, realizamos alguns experimentos de referência que são descritos a seguir.

Inicialmente, foram geradas 10.000 alocações aleatórias para cada variação do Gauss18, isto é, distribuições aleatórias das tarefas entre os processadores P0 e P1. Subseqüentemente, o tempo total de execução T para cada variação foi calculado usando a política de escalonamento “a tarefa com maior nível dinâmico é executada primeiro”. Os resultados estão apresentados na segunda coluna da Tabela 6.1, onde a média de T obtida nas 10.000 alocações para cada variação do Gauss18 é

apresentada.

Grafo	Experimentos de Referência		Evolução Simples				Evolução Conjunta			
	Alocações Aleatórias	T_{AG_Ref}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{min}	T_{max}
Gauss18_1	75,92	47,0	52,91	3,99	52,0	79,0	47,24	1,24	47,0	56,0
Gauss18_2	77,38	47,0	67,95	8,22	59,0	85,0	47,48	1,61	47,0	57,0
Gauss18_3	70,67	47,0	53,54	6,37	51,0	83,0	47,19	1,37	47,0	60,0
Gauss18_4	75,19	47,0	52,03	4,59	51,0	82,0	49,35	1,44	47,0	56,0
Gauss18_5	77,71	50,0	54,00	0,0	54,0	54,0	52,28	1,78	50,0	63,0
Gauss18_6	74,0	47,0	48,09	0,90	48,0	57,0	48,00	6,54	47,0	106,0
Gauss18_7	78,0	44,0	44,00	0,00	44,0	44,0	48,23	2,94	47,0	62,0
Gauss18_8	79,0	44,0	57,20	2,27	51,0	70,0	47,02	0,20	47,0	49,0
Gauss18_9	79,0	47,0	55,87	12,92	48,0	86,0	47,67	2,18	47,0	64,0
Gauss18_10	80,0	44,0	51,46	2,65	51,0	74,0	47,63	2,46	47,0	59,0
Gauss18_11	78,08	47,0	64,50	7,85	51,0	92,0	47,28	0,76	47,0	51,0
Gauss18_12	74,10	47,0	48,14	0,98	48,0	55,0	47,02	0,20	47,0	49,0
Gauss18_13	71,22	44,0	61,15	9,90	47,0	82,0	47,00	0,00	47,0	47,0
Gauss18_14	77,98	46,0	63,44	9,98	55,0	83,0	47,19	1,07	47,0	56,0
Gauss18_15	74,13	47,0	57,92	10,36	53,0	86,0	47,24	1,33	47,0	59,0
Média (10)	76,55	45,70	55,17	5,78	49,60	72,90	47,42	1,76	47,0	60,20
Média (15)	76,15	46,33	55,48	5,39	50,86	74,13	47,85	1,67	47,20	59,60

Tab. 6.1: Evolução Simples e Conjunta, utilizando-se o modo de operação sequencial, a Vizinhança selecionada e atributo nível dinâmico.

Conforme apresentamos na Seção 5.3, um AG foi implementado para escalonar diretamente as tarefas e aplicado para as quinze variações do Gauss18. Para facilitar a análise dos resultados, repetimos os resultados apresentados na Tabela 5.3 na terceira coluna da Tabela 6.1. Analisando-se a segunda e terceira colunas da Tabela 6.1, vê-se que, como esperado, os tempos encontrados utilizando-se o AG foram muito melhores que o tempo médio encontrado nas alocações aleatórias. Embora os melhores resultados encontrados pelo AG não possam ser assumidos como os valores ótimos para cada grafo de programa, eles podem ser usados como valores de referência de bons resultados a serem alcançados pelas regras de transição.

Baseado nestes resultados, a avaliação da capacidade de generalização das regras evoluídas foi realizada aplicando-as nas quinze variações do Gauss18, onde foram geradas 100 configurações iniciais aleatórias para testar cada regra. Os resultados são apresentados da quarta à sétima coluna (Evolução Simples) da Tabela 6.1, sendo T_{med} o tempo de execução médio obtido nas 100 CIs, T_{dp} o desvio padrão de T nas 100 CIs, T_{min} o menor tempo de execução e T_{max} o maior tempo de execução nas 100 CIs. Comparados aos resultados encontrados pelo AG de referência, somente em um único grafo (Gauss18_7), as regras de transição de ACs foram aptas a encontrar a melhor alocação em 100% dos testes. Nos outros 14 grafos, os resultados obtidos pelas regras foram piores que os resultados

obtidos pelo AG de referência. Entretanto, os resultados médios obtidos pelas regras de AC foram muito melhores que os valores médios obtidos com as alocações aleatórias. Assim, embora as regras obtidas usando-se somente o Gauss18 não retornem os melhores resultados possíveis quando as mesmas são aplicadas nas quinze variações, é possível concluir que estas regras possuem alguma capacidade de generalização. No entanto, há muito espaço para que a capacidade de generalização seja melhorada. Calculando-se o intervalo de confiança (Seção 2.11) [73] para os resultados obtidos através destas regras, nós estamos 95% confiantes de que a média do tempo total de execução T está no intervalo [54,42; 56,54] para as quinze variações, e no intervalo [54,04; 56,31] para as dez variações de Gauss18_6 a Gauss18_15.

Um novo experimento foi conduzido utilizando-se a nova abordagem Evolução Conjunta. Na fase de aprendizagem do algoritmo de escalonamento baseado em AC, seis grafos de programa foram utilizados para calcular o valor de aptidão de uma regra de transição. Uma regra capaz de escalonar os seis grafos de programa deve retornar um valor de aptidão igual a 282,0, de acordo com os melhores tempos encontrados pelo AG de referência. As regras com o menor valor de aptidão são consideradas mais gerais.

Foram realizadas trinta execuções do AG e, na melhor delas, uma média de T para o melhor indivíduo na última geração foi 291,40. A Figura 6.1 apresenta o processo evolutivo durante a fase de aprendizagem de regras na evolução conjunta. Na primeira geração dessa execução, a média de T para o melhor indivíduo é 386,52. A aptidão do melhor indivíduo oscila entre os valores 290,0 e 299,0, a partir da geração 30. O AG modifica as regras e, na geração 100, o valor 291,40 é alcançado.

Nós aplicamos a melhor regra evoluída nas 5 variações do Gauss18 usadas na fase de aprendizagem e também nas outras 10 variações que não participaram nesta fase. Os resultados estão apresentados nas últimas colunas (Evolução Conjunta) da Tabela 6.1. As regras evoluídas conjuntamente retornaram melhores resultados para os grafos de programa Gauss18_1, Gauss18_2, Gauss18_3, Gauss18_4 e Gauss18_5, quando comparadas às regras evoluídas para um único grafo, no que diz respeito ao tempo médio. Mais especificamente, em respeito aos melhores resultados encontrados nas 100 CIs avaliadas, elas podem ser consideradas tão boas quanto o AG de referência. Esta melhoria era previsível, uma vez que as informações destes grafos foram utilizadas na fase de aprendizagem. Além disso, foi possível perceber uma melhoria significativa na capacidade de generalização das re-

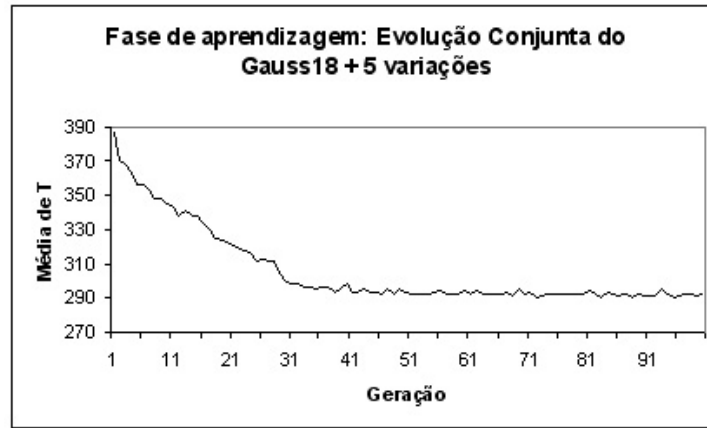


Fig. 6.1: Fase de aprendizagem do algoritmo de escalonamento para o Gauss18 e algumas de suas variações. Utilização da Vizinhaça Seleccionada e do atributo nível dinâmico.

gras evoluídas. As regras evoluídas conjuntamente, em geral, encontraram bons resultados também para as 10 variações do Gauss18 não utilizadas na fase de aprendizagem. Exceto para o Gauss18_7, os valores médios diminuíram significativamente, em relação às regras evoluídas com a Evolução Simples. Para cinco variações (Gauss18_6, Gauss18_9, Gauss18_11, Gauss18_12 e Gauss18_15) foram encontrados resultados equivalentes aos do AG de referência. Em outras três variações (Gauss18_8, Gauss18_10 e Gauss18_14), embora um resultado equivalente ao AG de referência não tenha sido encontrado, podemos identificar uma melhoria em respeito tanto ao melhor tempo obtido quanto para o tempo médio. Calculamos o intervalo de confiança para as regras de AC no experimento com a evolução conjunta e estamos 95% confiantes que a média do tempo total de execução T está no intervalo [47,52; 48,15] nas quinze variações, e no intervalo [47,08; 47,77] nas dez variações que não participaram da fase de aprendizagem. Comparando-se com os intervalos de confiança obtidos com a Evolução Simples, é possível confirmar a melhoria no experimento com a evolução conjunta. Também avaliamos o teste de hipótese nula (Seção 2.11) [73] para comparar os dois experimentos. Há evidências significativas de que os tempos de execução obtidos pelas regras descobertas no experimento da evolução conjunta são melhores que os obtidos pela evolução simples. Estamos 95% confiantes de que esta melhoria está entre 6,51 e 8,73 unidades de tempo, considerando-se as quinze variações, e está entre 6,56 e 8,93 unidades de tempo, considerando-se apenas as dez variações de Gauss18_6 à Gauss18_15. Entendemos que estes resultados qualificam as regras descobertas pela evolução conjunta como sendo mais gerais que aquelas obtidas pela evolução simples.

O tempo de processamento médio desses experimentos, na fase de aprendizagem, é 0,50 hora para a Evolução Simples e 1,5 hora para a Evolução Conjunta. Na fase de operação, em ambas as abordagens, o tempo médio de processamento do algoritmo é de 0,25 hora.

Os resultados acima apresentados foram publicados em [113], sendo este um artigo apresentado no Simpósio Brasileiro de Redes Neurais (SBRN), realizado em Outubro de 2006.

6.1.2 Vizinhança totalística e atributos co-nível, custo computacional e custo de comunicação

Na Seção 5.6.1, apresentamos os resultados obtidos para o Gauss18, através da utilização da Vizinhança Totalística e dos atributos co-nível estático, custo computacional e custo de comunicação para os conjuntos *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)*, respectivamente. Assim, as regras evoluídas para o Gauss18 também foram utilizadas no escalonamento de 15 variações, para avaliarmos sua capacidade de generalização. Nesses testes, as regras são submetidas a 100 configurações iniciais diferentes para cada grafo avaliado. Para se avaliar a capacidade de generalização das regras evoluídas, utilizamos, como experimentos de referência, os resultados apresentados na Tabela 6.1 e repetidos na segunda e terceira colunas da Tabela 6.2.

Com base nestes resultados utilizados como referência, o experimento com a evolução simples foi analisado. A Tabela 6.2 (Evolução Simples) apresenta os resultados relativos ao escalonamento das 100 CIs realizado pela melhor regra evoluída na fase de aprendizagem, para cada grafo avaliado. Apenas em 8 grafos (Gauss18_3, Gauss18_4, Gauss18_5, Gauss18_7, Gauss18_8, Gauss18_9, Gauss18_11 e Gauss18_15) as regras de ACs evoluídas foram capazes de encontrar a melhor alocação diretamente obtida pelo AG. Em todos os outros grafos, o resultado da generalização foi inferior ao obtido pelo AG de referência. Entretanto, os resultados médios obtidos pela generalização das regras foi bem abaixo dos valores médios obtidos com as alocações aleatórias, apresentados na Tabela 6.2. Assim, também é possível constatar que alguma capacidade de generalização as regras evoluídas possuem. Calculando-se o intervalo de confiança para essas regras, pode-se dizer, com 95% de confiança, que a média de T está no intervalo [50,68; 51,86] para os quinze grafos de programa, e no intervalo [49,81; 50,98] para os grafos Gauss18_6 à Gauss18_15. Comparando-se com os resultados apresentados na Seção 6.1.1, percebemos que esse modelo de vizinhança obteve regras com maior

capacidade de generalização.

Grafo	Experimentos de Referência		Evolução Simples				Evolução Conjunta			
	Alocações Aleatórias	T_{AG_Ref}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{med}	T_{max}
Gauss18_1	75,92	47,0	50,87	2,70	48,0	54,0	52,82	1,66	51,0	59,0
Gauss18_2	77,38	47,0	53,40	2,38	51,0	65,0	53,68	2,13	51,0	63,0
Gauss18_3	70,67	47,0	52,91	3,24	47,0	68,0	49,31	4,27	47,0	77,0
Gauss18_4	75,19	47,0	53,52	4,05	47,0	68,0	50,32	3,64	47,0	67,0
Gauss18_5	77,71	50,0	54,41	2,99	50,0	68,0	50,68	1,58	50,0	55,0
Gauss18_6	74,0	47,0	53,61	3,66	51,0	71,0	52,54	1,55	51,0	56,0
Gauss18_7	78,0	44,0	47,32	1,37	44,0	54,0	48,39	2,82	44,0	53,0
Gauss18_8	79,0	44,0	49,32	3,18	44,0	55,0	48,41	2,15	47,0	53,0
Gauss18_9	79,0	47,0	47,90	2,94	47,0	67,0	48,51	3,58	47,0	67,0
Gauss18_10	80,0	44,0	51,56	2,96	46,0	55,0	48,61	2,36	47,0	53,0
Gauss18_11	78,08	47,0	51,03	3,41	47,0	62,0	49,02	2,50	47,0	53,0
Gauss18_12	74,10	47,0	53,80	1,79	51,0	56,0	52,20	1,96	48,0	56,0
Gauss18_13	71,22	44,0	48,89	2,93	46,0	55,0	48,05	3,28	44,0	57,0
Gauss18_14	77,98	46,0	48,49	3,11	47,0	74,0	48,91	3,22	47,0	59,0
Gauss18_15	74,13	47,0	52,03	4,45	47,0	65,0	49,85	3,58	47,0	67,0
Média (10)	76,55	45,70	50,39	2,98	47,0	61,40	49,45	2,70	46,90	57,40
Média (15)	76,15	46,33	51,27	3,01	47,53	62,46	50,08	2,68	47,66	59,66

Tab. 6.2: Evolução Simples e Conjunta, considerando-se o modo de operação sequencial, a Vizinhança Totalística e os atributos co-nível, custo computacional e custo de comunicação

Um novo experimento foi elaborado utilizando-se a abordagem Evolução Conjunta. O escalonamento ótimo para os seis grafos retorna um tempo total igual a 282,0, de acordo com os melhores resultados obtidos com o AG de referência. Foram feitas 30 execuções do AG e, na melhor delas, o T médio do melhor indivíduo evolui de 353,84 para 305,16 (Figura 6.2).

Posteriormente, as regras evoluídas com a nova abordagem foram aplicadas no escalonamento das 10 variações do Gauss18 não utilizadas na fase de aprendizagem. Nesses testes, as regras evoluídas foram também submetidas a 100 configurações iniciais diferentes para cada grafo avaliado e os resultados são apresentados na Tabela 6.2 (Evolução Conjunta).

Comparando-se as regras obtidas pelas duas abordagens, simples e conjunta, podemos observar que em relação ao menor tempo obtido nas 100 CIs, em dez grafos os mesmos resultados foram alcançados. Em três grafos (Gauss18_1, Gauss18_8 e Gauss18_10) a evolução simples foi superior e, em outros dois (Gauss18_12 e Gauss18_13), a evolução conjunta encontrou tempos menores. Assim, podemos concluir que em relação a T_{min} os resultados foram muito similares. Entretanto, em relação ao tempo médio obtido nas alocações a partir de 100 CIs, a abordagem conjunta obteve resultados superiores em 10 dos 15 grafos analisados e em apenas 5 (Gauss18_1, Gauss18_2, Gauss18_7, Gauss18_9 e Gauss18_14) a abordagem simples foi melhor. Esses resultados são refletidos na média das médias

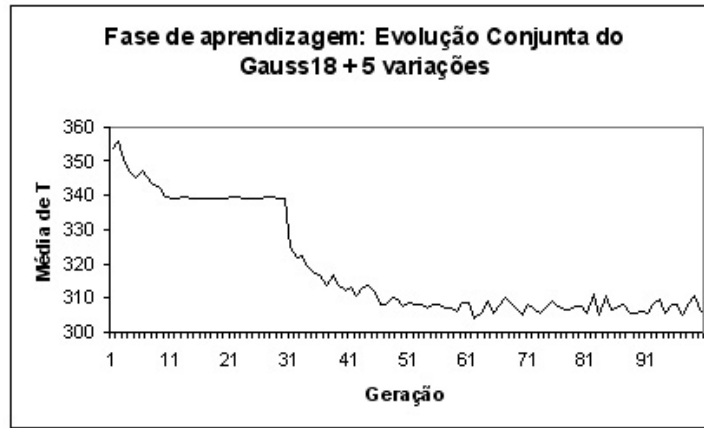


Fig. 6.2: Fase de aprendizagem do algoritmo de escalonamento: Evolução Conjunta considerando a Vizinhaça Totalística e os atributos co-nível estático, custo computacional e custo de comunicação.

obtidas em cada grafo que no caso da abordagem conjunta é 49,45 e no caso da abordagem simples é 50,39. Além disso, o desvio padrão médio no caso da evolução conjunta foi 2,70 e no caso da evolução simples foi 2,98. Mas, foi em relação ao pior tempo obtido nas 100 CIs (T_{max}) que a abordagem conjunta se mostrou mais eficiente: em 6 dos 10 grafos analisados, o pior tempo obtido pela evolução conjunta foi menor. Em outros 2 grafos (Gauss18_9 e Gauss18_12) os resultados foram equivalentes e apenas em 2 grafos (Gauss18_13 e Gauss18_15), a abordagem simples foi mais eficiente. Essa superioridade se reflete na média dos maiores tempos obtidos, que no caso da abordagem conjunta é 57,40 e na abordagem simples é 61,40. E, quando calculado o intervalo de confiança para as regras obtidas através da evolução conjunta, pode-se dizer, com 95% de confiança, que a média de T está entre [49,56; 50,61] para os quinze grafos de programa, e no intervalo [48,92; 49,97] para as dez variações que não participaram da fase de aprendizagem. Comparando-se o intervalo de confiança calculado para as duas abordagens, percebe-se uma redução da média de T para a evolução conjunta. Avaliando-se também a hipótese nula para comparar os experimentos, encontram-se evidências de que os tempos de execução obtidos na evolução conjunta são melhores que os obtidos na evolução simples. Pode-se dizer, com 95% de confiança, que a redução do tempo médio está entre 0,39 e 1,97 unidades de tempo, considerando-se as quinze variações, e entre 0,15 e 1,74 unidades de tempo, considerando-se as dez variações. Comparando-se estes resultados com os apresentados na Seção 6.1.1, podemos perceber que, embora os resultados apresentados para esse modelo tenham sido menos expressivos, a contribuição da evolução conjunta para a melhoria da capacidade de generalização

das regras de ACs foi observada em ambos os modelos de vizinhança.

O tempo de processamento médio desses experimentos, na fase de aprendizagem, é 0,17 hora para a Evolução Simples e 0,40 hora para a Evolução Conjunta. Na fase de operação, em ambas as abordagens, o tempo médio de processamento do algoritmo é de 0,034 hora.

Os resultados acima foram publicados em [112], e o artigo foi apresentado no Congresso Brasileiro de Automática (CBA), ocorrido em Outubro de 2006.

6.1.3 Vizinhança Seleccionada e atributos co-nível, custo computacional e custo de comunicação

Um terceiro experimento foi realizado utilizando a Vizinhança Seleccionada e, diferentemente do experimento descrito na Seção 6.1.1, três atributos diferentes para os conjuntos *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)*. Os atributos utilizados foram o co-nível estático, o custo computacional e o custo de comunicação. Devemos ressaltar que esse modelo foi empregado em todos os experimentos realizados por Seredynski e colegas com o Gauss18 [95, 96, 99].

Aplicamos as regras descobertas pela Evolução Simples na fase de operação do algoritmo de escalonamento para as quinze variações do Gauss18. Os resultados obtidos estão apresentados na Tabela 6.3, onde os resultados dos experimentos de referência são replicados. Apenas para o Gauss18_7 houve uma regra capaz de encontrar o escalonamento ótimo a partir dos 100 testes de problemas. Calculamos o intervalo de confiança para este experimento, e estamos 95% confiantes que o valor médio de T , nas quinze variações, está no intervalo [54,77; 55,79]. E, nas variações de Gauss18_6 a Gauss18_15, este valor está no intervalo [51,13; 51,97]. Embora as médias sejam mais altas do que as obtidas nos modelos testados anteriormente (Tabelas 6.1 e 6.2), mais uma vez os resultados médios são bem abaixo das inicializações aleatórias.

Posteriormente, utilizamos a abordagem Evolução Conjunta na fase de aprendizagem do algoritmo de escalonamento para esse modelo de vizinhança. A Figura 6.3 apresenta o gráfico da melhor evolução, onde o valor de aptidão do melhor indivíduo da população a cada geração é apresentado. Na primeira geração, a aptidão do melhor indivíduo é igual a 375,76. O AG modifica as regras de transição de AC (indivíduos) até que, na geração 69, é descoberta uma regra com o valor de aptidão 299,0, valor este que permanece até a última geração.

Grafo	Experimentos de Referência		Evolução Simples				Evolução Conjunta			
	Alocações Aleatórias	T_{AG_Ref}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{med}	T_{max}
Gauss18_1	75,92	47,0	53,67	2,57	53,0	67,0	49,00	0,00	49,0	49,0
Gauss18_2	77,38	47,0	62,49	5,08	59,0	76,0	52,00	0,00	52,0	52,0
Gauss18_3	70,67	47,0	65,04	6,51	56,0	76,0	49,00	0,00	49,0	49,0
Gauss18_4	75,19	47,0	66,23	0,55	63,0	67,0	49,00	0,00	49,0	49,0
Gauss18_5	77,71	50,0	66,26	2,94	63,0	91,0	51,00	0,00	51,0	51,0
Gauss18_6	74,0	47,0	48,00	0,00	48,0	48,0	53,00	0,00	53,0	53,0
Gauss18_7	78,0	44,0	44,00	0,00	44,0	44,0	47,00	0,00	47,0	47,0
Gauss18_8	79,0	44,0	54,41	3,04	53,0	62,0	47,00	0,00	47,0	47,0
Gauss18_9	79,0	47,0	49,05	4,17	48,0	71,0	49,00	0,00	49,0	49,0
Gauss18_10	80,0	44,0	54,05	1,68	53,0	62,0	48,98	0,20	47,0	49,0
Gauss18_11	78,08	47,0	51,17	3,94	47,0	62,0	49,00	0,00	49,0	49,0
Gauss18_12	74,10	47,0	48,00	0,00	48,0	48,0	51,00	0,00	51,0	51,0
Gauss18_13	71,22	44,0	53,48	1,79	53,0	61,0	45,00	0,00	45,0	45,0
Gauss18_14	77,98	46,0	58,65	5,23	55,0	72,0	48,00	0,00	48,0	48,0
Gauss18_15	74,13	47,0	54,71	1,49	53,0	56,0	49,00	0,00	49,0	49,0
Média (10)	76,55	45,70	51,55	2,13	50,20	58,60	48,69	0,02	48,50	48,70
Média (15)	76,15	46,33	55,28	2,59	53,06	64,20	49,13	0,01	49,00	49,13

Tab. 6.3: Evolução Simples e Conjunta, considerando-se o modo de operação seqüencial, a Vizinhança Seleccionada e os atributos co-nível, custo computacional e custo de comunicação.

Aplicamos a melhor regra evoluída nas 15 variações do Gauss18. Os resultados estão apresentados nas últimas colunas (Evolução Conjunta) da Tabela 6.3. As regras evoluídas conjuntamente retornaram melhores resultados para os grafos de programa Gauss18_1, Gauss18_2, Gauss18_3, Gauss18_4 e Gauss18_5, no que diz respeito ao tempo médio. Lembramos que as informações destes grafos foram utilizadas na fase de aprendizagem e uma redução no tempo dos mesmos era esperada. As regras evoluídas conjuntamente, em geral, encontraram bons resultados também para as 10 variações do Gauss18 não utilizadas na fase de aprendizagem. Exceto para os grafos Gauss18_6, Gauss18_7 e Gauss18_12, os valores médios diminuíram significativamente e, embora um resultado equivalente ao AG de referência não tenha sido encontrado para nenhuma das variações, podemos identificar uma melhoria em respeito tanto ao melhor tempo obtido quanto ao tempo médio. Calculamos o intervalo de confiança para as regras descobertas por Evolução Conjunta e estamos 95% confiantes que a média de T está no intervalo [49,12; 49,13] para as quinze variações, e no intervalo [48,69; 48,70] para as dez variações que não participaram da fase de aprendizagem. Comparando-se os intervalos de confiança entre as abordagens Simples e Conjunta, é possível confirmar a melhoria no segundo experimento. Através do teste de hipótese nula verificamos que há evidências significativas de que os tempos de execução retornados pelas regras descobertas pela Evolução Conjunta são menores que os obtidos no experimento da Evolução Simples. Estamos 95% confiantes de que esta

melhoria está entre 5,63 e 6,65 unidades de tempo, considerando-se as quinze variações, e entre 2,43 e 3,27 unidades de tempo, considerando-se as dez variações de Gauss18_6 a Gauss18_15. Entendemos que estes resultados qualificam as regras descobertas pela Evolução Conjunta como sendo mais gerais que aquelas obtidas pela Evolução Simples.

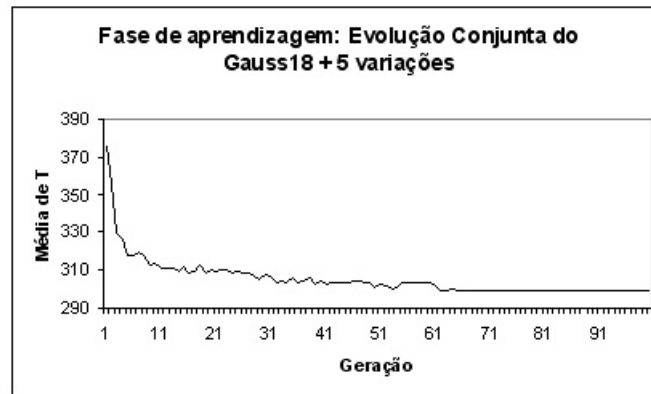


Fig. 6.3: Fase de aprendizagem do algoritmo de escalonamento utilizando-se a abordagem Evolução Conjunta: Gauss18 + 5 variações, Vizinhança Seleccionada e três atributos diferentes para a formação da vizinhança.

Comparando-se os resultados com o experimento descrito na Seção 6.1.1, onde utilizamos apenas o atributo nível dinâmico na vizinhança selecionada, percebemos que os resultados descritos acima foram um pouco inferiores. Porém, independentemente do(s) atributo(s) utilizado(s) na vizinhança selecionada, a evolução conjunta mostrou-se superior à evolução simples, descobrindo regras de transição de AC com uma maior capacidade de generalização. Além disso, é importante ressaltar que o tempo de processamento do algoritmo de escalonamento é menor quando se usa os atributos co-nível estático, custo computacional e custo de comunicação. Estes atributos têm seu valor invariável, para o grafo de programa, durante a execução do algoritmo. Já o atributo nível dinâmico, que é calculado em função da alocação das tarefas nos processadores, deve ser calculado sempre que uma tarefa é alocada em um processador diferente do qual ela estava alocada. Logo, na evolução do AC, quando o atributo nível dinâmico é utilizado na formação da vizinhança, a cada mudança de uma tarefa de processador o atributo deverá ser calculado novamente para cada tarefa que antecede a tarefa atualizada (inclusive a própria tarefa). Este cálculo pode ser efetuado diversas vezes, aumentando-se assim o tempo de processamento do algoritmo.

O tempo de processamento médio desses experimentos, na fase de aprendizagem, é 0,20 hora

para a Evolução Simples e 0,53 hora para a Evolução Conjunta. Na fase de operação, em ambas as abordagens, o tempo médio de processamento do algoritmo é de 0,05 hora.

6.1.4 Vizinhança Totalística e atributo nível dinâmico

O último experimento que realizamos com o Gauss18 utilizou o modelo de vizinhança totalística e apenas o atributo nível dinâmico para a formação da vizinhança. Aplicamos as regras descobertas na Evolução Simples na fase de operação para as quinze variações do Gauss18, utilizando-se 100 CIs para cada variação. Os resultados obtidos estão apresentados na Tabela 6.4 (Evolução Simples), onde também foram replicados os experimentos de referência. Para todas as variações do Gauss18, não houve regra capaz de encontrar, na média, o escalonamento ótimo a partir dos 100 testes de problemas criados. Calculamos o intervalo de confiança para os valores obtidos neste experimento e, estamos 95% confiantes de que o valor médio de T está no intervalo [49,42; 50,30] para as quinze variações, e no intervalo [47,79; 48,70] para as dez variações de Gauss18_6 à Gauss18_15.

Grafo	Experimentos de Referência		Evolução Simples				Evolução Conjunta			
	Alocações Aleatórias	T_{AG_Ref}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{med}	T_{max}
Gauss18_1	75,92	47,0	49,87	1,89	48,0	57,0	53,07	2,05	49,0	58,0
Gauss18_2	77,38	47,0	52,73	2,85	47,0	57,0	52,24	2,88	49,0	65,0
Gauss18_3	70,67	47,0	53,30	3,92	47,0	68,0	51,58	2,01	49,0	55,0
Gauss18_4	75,19	47,0	54,51	0,95	50,0	55,0	51,99	1,87	49,0	61,0
Gauss18_5	77,71	50,0	55,07	0,66	52,0	56,0	54,40	2,20	52,0	61,0
Gauss18_6	74,0	47,0	48,21	1,83	47,0	56,0	51,72	2,37	47,0	55,0
Gauss18_7	78,0	44,0	44,70	1,66	44,0	51,0	48,03	2,41	44,0	57,0
Gauss18_8	79,0	44,0	48,80	2,72	47,0	59,0	49,68	2,00	47,0	56,0
Gauss18_9	79,0	47,0	47,67	0,89	47,0	52,0	48,48	1,94	47,0	51,0
Gauss18_10	80,0	44,0	48,39	4,73	44,0	76,0	50,13	2,19	44,0	55,0
Gauss18_11	78,08	47,0	49,01	2,72	47,0	62,0	49,80	1,93	47,0	55,0
Gauss18_12	74,10	47,0	48,26	1,71	47,0	55,0	52,11	2,34	47,0	55,0
Gauss18_13	71,22	44,0	48,60	2,07	47,0	53,0	49,53	1,76	47,0	54,0
Gauss18_14	77,98	46,0	49,04	3,02	48,0	66,0	49,11	2,16	47,0	66,0
Gauss18_15	74,13	47,0	49,75	1,72	47,0	53,0	50,34	2,92	47,0	61,0
Média (10)	76,55	45,70	48,24	2,30	46,50	58,30	49,89	2,20	46,40	56,50
Média (15)	76,15	46,33	49,86	2,22	47,26	58,40	50,81	2,20	47,46	57,66

Tab. 6.4: Evolução Simples e Conjunta do Gauss18, considerando-se o modo de operação sequencial, a Vizinhança Totalística e o atributo nível dinâmico.

Posteriormente, utilizamos a abordagem Evolução Conjunta na fase de aprendizagem do algoritmo de escalonamento. A Figura 6.4 apresenta a melhor execução desse experimento, onde o valor de aptidão do melhor indivíduo da população a cada geração é representado no gráfico. Na primeira geração, a aptidão do melhor indivíduo é igual a 363,12. O AG modifica as regras de transição de

AC (indivíduos) até a última, onde a melhor regra da população apresentou o valor de aptidão médio igual a 312,68.

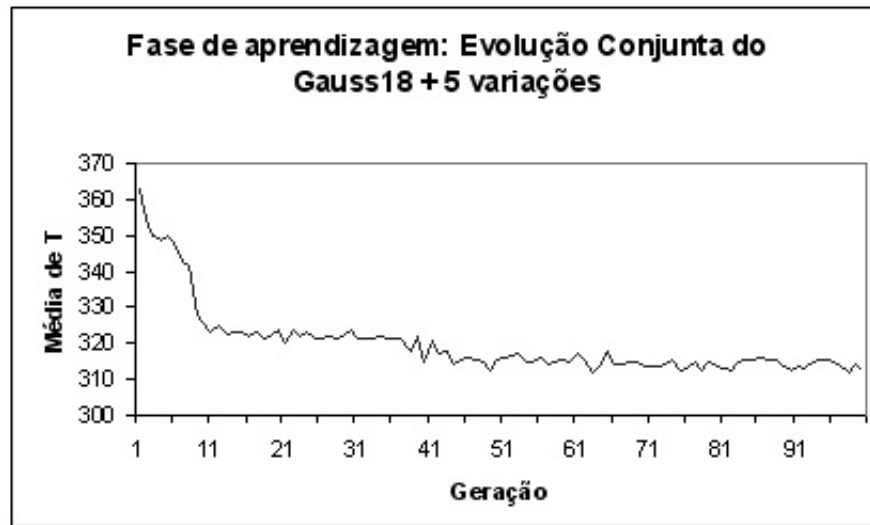


Fig. 6.4: Fase de aprendizagem do algoritmo de escalonamento utilizando-se a abordagem Evolução Conjunta: Gauss18 + 5 variações, Vizinhança Totalística e atributo nível dinâmico para a formação da vizinhança.

Aplicamos a melhor regra evoluída nas 5 variações do Gauss18 usadas na fase de aprendizagem e também nas outras 10 variações que não participaram nesta fase. Os resultados estão apresentados nas últimas colunas (Evolução Conjunta) da Tabela 6.4. As regras evoluídas conjuntamente retornaram melhores resultados apenas para os grafos de programa Gauss18_2, Gauss18_3, Gauss18_4, Gauss18_5, quando comparadas às regras evoluídas para um único grafo, no que diz respeito ao tempo médio. Em todos os outros grafos a Evolução Simples foi melhor. Com relação ao melhor tempo (T_{min}), em 10 grafos de programa houve empate, em 2 a Evolução Conjunta foi melhor e em 3 a Evolução Simples obteve melhores tempos. Porém, observando os resultados obtidos para os 10 grafos que não participaram da fase de aprendizagem (Gauss18_6 a Gauss18_15), observa-se que o desvio padrão (T_{dp}), o melhor (T_{min}) e o pior tempo (T_{max}) apresentaram tempos melhores com a Evolução Conjunta. Calculamos o intervalo de confiança para as regras de AC descobertas neste experimento e estamos 95% confiantes que a média do tempo total de execução T para as regras descobertas com a Evolução Conjunta está no intervalo [50,38; 51,24] para as quinze variações e [49,46; 50,32] para as dez variações que não participaram da fase de aprendizagem. Avaliando-se o teste de hipótese nula, tem-se que a Evolução Simples foi melhor que a Evolução Conjunta entre 0,34 e

1,56 unidades de tempo, considerando-se as quinze variações, e entre 1,02 e 2,27 unidades de tempo, considerando-se as dez variações de Gauss18_6 a Gauss18_15. Neste caso, pudemos perceber que os resultados obtidos com a Evolução Simples foram um pouco melhores que os obtidos com a Evolução Conjunta. Logo, a utilização da Evolução Conjunta no modelo de vizinhança totalística com o atributo nível dinâmico não contribuiu para uma melhor capacidade de generalização das regras. Ao contrário, a Evolução Simples mostrou-se superior. Portanto, dos 4 modelos de vizinhança para o Gauss18, apenas nesse modelo a Evolução Conjunta não se mostrou eficaz. Também cabe ressaltar que de todos os modelos, esse foi o que retornou o melhor resultado de generalização com a Evolução Simples: média de 49,86 nas quinze variações do Gauss18 (55,48 na Vizinhança Seleccionada com o atributo nível dinâmico, 51,27 na Vizinhança Totalística com três atributos e 55,28 na Vizinhança Seleccionada com três atributos). Esse resultado só é inferior ao obtido com a Evolução Conjunta no modelo Vizinhança Seleccionada com o atributo nível dinâmico (47,85) e no modelo Vizinhança Seleccionada com três atributos (49,13). Assim, concluímos que esse é o modelo, dentre os avaliados, que possui melhor capacidade de generalização na Evolução Simples por um lado, mas que não foi beneficiado pela Evolução Conjunta, por outro lado. Ressaltamos que não existe relato na literatura da utilização do atributo nível dinâmico na formação da vizinhança totalística para o Gauss18, ou qualquer outro grafo.

O tempo de processamento médio desses experimentos, na fase de aprendizagem, é 0,76 hora para a Evolução Simples e 1,60 hora para a Evolução Conjunta. Na fase de operação, em ambas as abordagens, o tempo médio de processamento do algoritmo é de 0,21 hora.

6.2 Aprendizagem coevolutiva: uma nova abordagem na busca evolutiva de regras de AC

Com o objetivo de melhorar ainda mais a capacidade de generalização das regras de transição, propusemos uma outra abordagem para a fase de aprendizagem, onde utilizamos um Algoritmo Genético Coevolutivo (AGC) ao invés de um AG Padrão. Os algoritmos genéticos coevolutivos (AGCs) [48] diferem dos algoritmos genéticos tradicionais, por utilizarem duas populações durante o processo evolutivo: uma de possíveis soluções e outra de instâncias do problema a resolver. Em nossa aborda-

gem, as duas populações do AGC são: a população *Pop1* de possíveis soluções para o problema, isto é, regras de transição de AC, e a população *Pop2* composta por diferentes grafos de programa.

O principal grafo de programa utilizado nos experimentos foi o Gauss18. Os grafos de programa utilizados para avaliar a capacidade de generalização das regras de transição de AC foram cinquenta variações do Gauss18 geradas aleatoriamente. Para as variações do grafo de programa, as possíveis modificações apresentadas são: (i) alteração no custo computacional de uma determinada tarefa, (ii) alteração no custo de comunicação entre tarefas k e l , onde k precede l , (iii) inclusão de uma nova aresta entre duas tarefas k e l , onde k não precede l e o número de ordem (identificação da tarefa) da tarefa k é menor que o número de ordem da tarefa l , (iv) exclusão de uma aresta entre duas tarefas k e l , onde k precede l . Uma variação do Gauss18 pode possuir uma ou mais modificações em relação ao original, de diferentes tipos (i, ii, iii e iv). As cinquenta variações do Gauss18 utilizadas na avaliação das regras descobertas na fase de aprendizagem do algoritmo de escalonamento tiveram sua nomenclatura assim definida: os grafos de programa V18_10 a V18_19 possuem, cada um, apenas 1 modificação em relação ao Gauss18, os grafos V18_20 a V18_29 possuem 2 modificações, os grafos V18_30 a V18_39 possuem 3 modificações, os grafos V18_40 a V18_49 possuem 4 modificações e os grafos V18_50 a V18_59 possuem 5 modificações.

A população *Pop2* é composta, além do Gauss18, por diferentes variações aleatórias deste grafo de programa, sendo que estes grafos de programa podem conter de 1 a 4 modificações, dentre as acima citadas, em relação ao grafo de programa original.

No AGC, durante o processo evolutivo, ambas as populações são alteradas por operadores genéticos. Esse tipo de estratégia coevolutiva é chamada de coevolução total [48]. Na população *Pop1*, são aplicados os operadores de *seleção*, *crossover* e *mutação* para evoluir as regras de transição. Na população *Pop2*, apenas os operadores de *seleção* e *mutação* são utilizados para, respectivamente, selecionar grafos de programa para deixar a população e inserir novas variações do Gauss18.

Para a população de regras de transição (*Pop1*), os parâmetros do AG e do AC utilizados foram: $P = 100$, $E = 10$, $p_c = 0.50$, $p_m = 0.012$, $G = 100$, $CIs = 25$ e $M = 50$. O processo de seleção para reprodução utiliza uma estratégia elitista, isto é, pares de indivíduos são escolhidos aleatoriamente da elite para gerar novos filhos, o operador de *crossover* utilizado foi o de ponto-simples e o operador de *mutação* foi o de complemento do alelo.

A população *Pop2* inicial é composta pelo Gauss18 e mais 9 grafos de programa gerados aleatoriamente, cada um representando uma variação do Gauss18, podendo apresentar de 1 a 4 modificações em relação ao original. A cada geração, são utilizados 10 grafos de programa diferentes para se avaliar as regras de transição de AC da população *Pop1*. Foram utilizadas 25 configurações iniciais aleatórias, representando alocações das tarefas dos grafos de programa nos processadores P0 e P1, e cada regra de *Pop1* é testada nas 25 CIs para cada grafo de programa da população *Pop2*. O valor de aptidão da regra é o somatório de T encontrado a partir da aplicação da mesma em cada grafo de programa de *Pop2*, partindo-se de cada uma das 25 CIs. O valor de aptidão de um grafo de programa da população *Pop2* é o somatório de T obtido em cada CI aplicando-se cada regra da população *Pop1* no grafo em questão. A população *Pop2*, durante o processo coevolutivo, é modificada pelos operadores de *seleção* e *mutação*. A estratégia de seleção que adotamos para a *Pop2* foi a seguinte: para cada grafo de programa, é dada a “oportunidade” de participar do processo evolutivo por, no mínimo, 10 gerações. Assim, nas primeiras 10 gerações do AGC, os 10 grafos de programa gerados aleatoriamente na primeira geração permanecem inalterados. A partir da décima geração, a cada geração um grafo de programa é selecionado para deixar a população, dando lugar a outro grafo de programa que é uma variação do Gauss18 gerada aleatoriamente. Esta variação, que representa uma mutação do Gauss18, pode possuir de 1 a 4 modificações em relação ao grafo de programa original. O grafo de programa selecionado para deixar a população é aquele que já participou de 10 ou mais gerações e que possui menor valor de aptidão.

Realizamos um primeiro experimento utilizando o modelo de vizinhança selecionada e os atributos co-nível estático, custo computacional e custo de comunicação para os conjuntos de *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)*. Optamos por iniciar os experimentos utilizando o modelo de vizinhança selecionada por ele ter apresentado resultados melhores que a vizinhança totalística para o Gauss18 na Evolução Conjunta, conforme apresentamos na Seção 5.6. Quanto à escolha dos três atributos para a formação da vizinhança, o tempo de processamento do algoritmo é menor quando os utilizamos, ao invés de utilizar apenas o atributo nível dinâmico. Esse novo modelo de evolução exigiu inúmeros experimentos preliminares para ajustes dos diversos parâmetros e estratégias envolvidas na coevolução. Assim, a opção por um modelo de vizinhança que exigisse um menor esforço computacional foi extremamente importante para viabilizar os experimentos com a nova abordagem.

Na Seção 6.1.3, apresentamos o experimento com a abordagem Evolução Conjunta e o modelo de vizinhança acima citado, onde as regras descobertas mostraram-se mais gerais quando comparadas às regras descobertas com a Evolução Simples. Antes de realizarmos o experimento com o AGC, aplicamos estas regras descobertas com a Evolução Conjunta no escalonamento das cinquenta variações aleatórias do Gauss18 para, novamente, testar sua capacidade de generalização. A Tabela 6.5 apresenta os resultados obtidos para a melhor regra da população para cada uma das cinquenta variações. As médias gerais, isto é, nas 50 variações, foram $T_{med_geral} = 53,63$, $T_{dp_geral} = 3,53$, $T_{min_geral} = 50,22$ e $T_{max_geral} = 64,68$. Calculando-se o intervalo de confiança para os resultados obtidos através destas regras evoluídas para o Gauss18 e as cinco variações (Gauss18_1 a Gauss18_5) no escalonamento das cinquenta variações aleatórias, estamos 95% confiantes de que a média do tempo total de execução T está no intervalo [52,94; 54,32].

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,0	0,00	49,0	49,0	V18_20	53,06	4,89	50,0	74,0
V18_11	53,0	0,00	53,0	53,0	V18_21	60,40	4,88	52,0	72,0
V18_12	49,0	0,00	49,0	49,0	V18_22	49,0	0,00	49,0	49,0
V18_13	49,0	0,00	49,0	49,0	V18_23	56,46	6,40	49,0	68,0
V18_14	48,98	0,20	47,0	49,0	V18_24	51,0	0,00	51,0	51,0
V18_15	49,0	0,00	49,0	49,0	V18_25	57,50	5,55	52,0	68,0
V18_16	48,98	0,20	48,0	49,0	V18_26	56,42	4,53	52,0	62,0
V18_17	49,0	0,00	49,0	49,0	V18_27	48,98	0,20	47,0	49,0
V18_18	49,0	0,00	49,0	49,0	V18_28	57,53	5,41	53,0	69,0
V18_19	54,95	3,53	49,0	59,0	V18_29	58,66	6,16	53,0	78,0
Média	49,99	0,39	49,20	50,40	Média	54,90	3,80	50,80	64,0
V18_30	57,87	6,22	49,0	71,0	V18_40	58,38	10,86	49,0	86,0
V18_31	49,0	0,00	49,0	49,0	V18_41	61,71	2,70	61,0	74,0
V18_32	56,37	6,65	50,0	75,0	V18_42	52,97	5,90	47,0	66,0
V18_33	48,60	6,17	46,0	80,0	V18_43	50,20	2,09	48,0	53,0
V18_34	53,75	4,51	49,0	70,0	V18_44	52,41	3,45	50,0	84,0
V18_35	46,0	0,00	46,0	46,0	V18_45	59,12	8,42	47,0	78,0
V18_36	54,0	0,00	54,0	54,0	V18_46	54,44	3,61	52,0	73,0
V18_37	49,25	1,43	49,0	58,0	V18_47	55,17	4,02	50,0	70,0
V18_38	49,62	4,00	47,0	66,0	V18_48	63,57	7,18	52,0	73,0
V18_39	56,26	7,40	49,0	72,0	V18_49	60,36	6,54	53,0	83,0
Média	52,07	3,63	48,80	64,10	Média	56,83	5,47	50,90	74,0
V18_50	55,57	3,29	55,0	76,0	Média geral	53,63	3,53	50,22	64,68
V18_51	57,06	3,13	53,0	61,0					
V18_52	47,18	3,39	46,0	72,0					
V18_53	60,55	5,34	56,0	81,0					
V18_54	51,79	6,60	49,0	77,0					
V18_55	52,66	3,57	49,0	58,0					
V18_56	49,0	0,00	49,0	49,0					
V18_57	56,56	6,00	54,0	83,0					
V18_58	49,88	3,62	49,0	69,0					
V18_59	63,60	8,59	54,0	83,0					
Média	54,38	4,35	51,40	70,90					

Tab. 6.5: Aplicação das regras obtidas com a Evolução Conjunta, Vizinhança Seleccionada e três atributos diferentes, no escalonamento das 50 variações aleatórias do Gauss18.

No próximo experimento realizado, utilizamos o AGC para a fase de aprendizagem ao invés do AG Padrão. A Tabela 6.6 apresenta os resultados obtidos na avaliação das regras co-evoluídas nas cinquenta variações aleatórias do Gauss18. A Figura 6.5 apresenta a aptidão média do melhor indivíduo das populações *Pop1* e *Pop2*, ao longo das gerações de uma execução do AGC. Na primeira geração, a aptidão média do melhor indivíduo da população de regras (*Pop1*) é 61,65 e na população de grafos de programa (*Pop2*) é 78,96. Em *Pop1*, a aptidão média do melhor indivíduo tende a diminuir com o passar das gerações, uma vez que as melhores regras (menor valor médio de T) são escolhidas pelo AG para permanecerem e gerar filhos, alcançando o valor 48,10 na última geração. Em *Pop2*, a partir da geração 10, a aptidão média do melhor indivíduo oscila entre 45 e 65 até que, na última geração, este valor é 51,07.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	45,59	1,50	44,0	47,0	V18_20	47,22	2,85	44,0	59,0
V18_11	48,58	1,00	48,0	56,0	V18_21	50,69	7,73	44,0	75,0
V18_12	44,16	1,12	44,0	52,0	V18_22	49,33	3,89	44,0	61,0
V18_13	45,20	1,66	44,0	52,0	V18_23	48,45	1,53	47,0	54,0
V18_14	45,07	1,57	44,0	52,0	V18_24	47,10	0,43	47,0	49,0
V18_15	44,21	1,22	44,0	52,0	V18_25	50,26	7,98	47,0	93,0
V18_16	44,0	0,00	44,0	44,0	V18_26	46,40	2,08	44,0	52,0
V18_17	44,35	1,65	44,0	55,0	V18_27	45,07	1,57	44,0	52,0
V18_18	47,87	9,99	44,0	94,0	V18_28	45,13	3,55	44,0	63,0
V18_19	46,59	2,21	44,0	55,0	V18_29	49,87	3,19	49,0	77,0
Média	45,56	2,19	44,40	55,90	Média	47,95	3,48	45,40	63,50
V18_30	45,81	2,90	44,0	58,0	V18_40	47,0	2,17	46,0	58,0
V18_31	44,35	1,65	44,0	55,0	V18_41	56,80	5,00	53,0	65,0
V18_32	48,80	3,12	47,0	61,0	V18_42	47,92	9,64	40,0	58,0
V18_33	44,77	3,84	44,0	79,0	V18_43	49,62	4,35	47,0	65,0
V18_34	47,0	0,00	47,0	47,0	V18_44	47,45	4,99	43,0	71,0
V18_35	47,16	1,61	45,0	63,0	V18_45	46,20	1,15	46,0	54,0
V18_36	49,39	1,86	49,0	66,0	V18_46	49,05	3,58	45,0	76,0
V18_37	53,17	3,20	47,0	59,0	V18_47	49,32	1,97	47,0	56,0
V18_38	48,84	1,45	44,0	57,0	V18_48	52,93	4,11	49,0	59,0
V18_39	50,81	3,91	44,0	59,0	V18_49	56,59	3,07	50,0	63,0
Média	48,01	2,35	45,50	60,40	Média	50,28	4,00	46,60	62,50
V18_50	53,64	4,72	50,0	66,0	Média geral	48,58	3,52	45,80	62,77
V18_51	51,72	7,92	46,0	69,0					
V18_52	45,78	4,42	43,0	83,0					
V18_53	52,48	3,34	51,0	63,0					
V18_54	51,22	5,25	47,0	68,0					
V18_55	52,59	9,82	45,0	89,0					
V18_56	50,83	3,86	47,0	62,0					
V18_57	49,97	2,85	49,0	77,0					
V18_58	52,05	11,48	44,0	83,0					
V18_59	51,17	2,15	49,0	56,0					
Média	51,14	5,58	47,10	71,60					

Tab. 6.6: Valores obtidos para 50 variações do Gauss18: abordagem do algoritmo genético coevolutivo.

Analisando-se os resultados apresentados nas Tabelas 6.5 e 6.6, pode ser visto que, dos 50 grafos

de programa utilizados para avaliar a capacidade de generalização das regras de transição de AC, em 45 os valores médios de T (coluna T_{med}) foram reduzidos com a utilização do algoritmo genético coevolutivo. Apenas para os grafos de programa V18_22, V18_35, V18_37, V18_56 e V18_58 os tempos médios de T obtidos pelas regras descobertas por Evolução Conjunta foram melhores do que os tempos obtidos no experimento com o AGC. Em todas as variações, o valor mínimo (coluna T_{min}) encontrado foi menor quando utilizada a abordagem de AGC. As Tabelas 6.5 e 6.6 também apresentam as médias parciais para cada grupo de 10 variações. Em todas as médias parciais, o AGC se mostrou superior ao AG com Evolução Conjunta. Calculando-se as médias gerais, tem-se que $T_{med_geral} = 48,58$, $T_{dp_geral} = 3,52$, $T_{min_geral} = 45,80$ e $T_{max_geral} = 62,77$. Como pode ser observado, em todas as médias gerais, os valores obtidos com o AGC foram menores do que os obtidos com a Evolução Conjunta, sendo as reduções mais significativas ocorridas para T_{med_geral} e T_{min_geral} . Calculamos o intervalo de confiança para as regras de AC no experimento com o AGC e estamos 95% confiantes que a média do tempo total de execução T está no intervalo $[47,90; 49,28]$. Comparando-se com o intervalo de confiança obtida para o experimento com a Evolução Conjunta $([52,94; 54,32])$, é possível confirmar a melhoria no experimento com o AGC. Nós também avaliamos o teste de hipótese nula para comparar os dois experimentos. Há evidências significativas de que os tempos de execução retornados pelas regras descobertas no experimento com o AGC são melhores que os obtidos no experimento da evolução conjunta. Estamos 95% confiantes de que esta melhoria está entre 3,96 e 6,12 unidades de tempo. Entendemos que estes resultados qualificam as regras descobertas com a abordagem coevolutiva como sendo mais gerais do que aquelas obtidas pela evolução conjunta.

6.2.1 Evolução Conjunta versus Coevolução

Após a realização dos testes com o modelo de vizinhança selecionada e os três atributos diferentes para a formação desta vizinhança, realizamos experimentos com o AGC utilizando a Vizinhança Selecionada e o atributo nível dinâmico, a Vizinhança Totalística e os três atributos diferentes, e a Vizinhança Totalística e o atributo nível dinâmico. Além disso, aplicamos as regras descobertas com a Evolução Conjunta apresentadas nas Seções 6.1.1, 6.1.2 e 6.1.4, para o escalonamento das 50 variações do Gauss18. As médias gerais para estes experimentos estão listadas nas últimas colunas da Tabela 6.7. As tabelas com os resultados detalhados são apresentadas no Apêndice D. Como pode

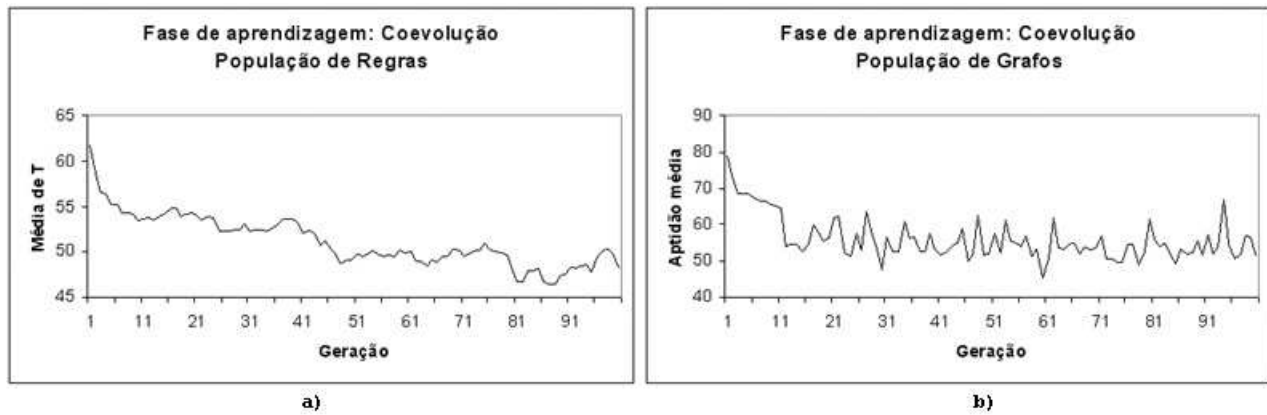


Fig. 6.5: Fase de aprendizagem do algoritmo de escalonamento baseado em ACs: processo coevolutivo das populações de regras e grafos de programa.

ser observado, considerando-se o parâmetro T_{med_geral} , apenas no modelo Vizinhaça Totalística e 3 atributos a Evolução Conjunta mostrou-se um pouco melhor que a abordagem coevolutiva, em relação à média. Entretanto, nesse modelo o mínimo geral com a Evolução Conjunta foi um pouco pior que a coevolução. Assim, consideramos os resultados dos experimentos muito similares. Nos demais modelos, o AGC apresentou melhores resultados.

Modelo	Evolução Conjunta	AGC
Vizinhaça Seleccionada 3 atributos	$T_{med_geral} = 53,6$ $T_{dp_geral} = 3,5$ $T_{min_geral} = 50,2$ $T_{max_geral} = 64,7$	$T_{med_geral} = 48,6$ $T_{dp_geral} = 3,5$ $T_{min_geral} = 45,8$ $T_{max_geral} = 62,8$
Vizinhaça Seleccionada Atributo nível dinâmico	$T_{med_geral} = 52,2$ $T_{dp_geral} = 5,2$ $T_{min_geral} = 47,5$ $T_{max_geral} = 71,4$	$T_{med_geral} = 49,3$ $T_{dp_geral} = 2,9$ $T_{min_geral} = 46,6$ $T_{max_geral} = 60,0$
Vizinhaça Totalística 3 atributos	$T_{med_geral} = 50,9$ $T_{dp_geral} = 2,7$ $T_{min_geral} = 46,8$ $T_{max_geral} = 57,7$	$T_{med_geral} = 51,1$ $T_{dp_geral} = 2,6$ $T_{min_geral} = 46,3$ $T_{max_geral} = 58,0$
Vizinhaça Totalística Atributo nível dinâmico	$T_{med_geral} = 51,1$ $T_{dp_geral} = 2,3$ $T_{min_geral} = 47,0$ $T_{max_geral} = 56,5$	$T_{med_geral} = 49,3$ $T_{dp_geral} = 2,8$ $T_{min_geral} = 46,4$ $T_{max_geral} = 56,5$

Tab. 6.7: Aplicação das regras descobertas com a Evolução Conjunta e AGC no escalonamento das 50 variações aleatórias do Gauss18.

Analisando-se a Tabela 6.7, fica claro que o experimento que resultou em melhorias mais significativas com a adoção da estratégia coevolutiva foi justamente o que utiliza o modelo de vizinhaça seleccionada e três atributos, por onde iniciamos nossos experimentos. Esse resultado mais expressivo

deve-se a dois motivos: primeiro, esse foi o modelo que retornou os piores resultados com a evolução conjunta e segundo porque esse foi o modelo que retornou os melhores resultados com a coevolução, e que também representa o melhor experimento de todos relatados até aqui, na generalização das cinquenta variações do Gauss18.

Creditamos esse resultado a dois fatores:

1. Esse modelo de vizinhança parece ser bem ajustado ao Gauss18. Nos experimentos iniciais que relatamos na Seção 5.6.1, dos quatro modelos analisados, esse foi o que de fato foi mais eficiente na busca de regras que resolvessem o Gauss18 original. Outra evidência desse ajustamento é o fato de que em todos os artigos publicados por Seredynski e colegas o modelo utilizado é sempre esse. Embora não justificado, em nenhum dos artigos os autores relatam a utilização da Vizinhança Totalística para o Gauss18, provavelmente por não ter apresentado resultados satisfatórios.
2. Não podemos esquecer que o ajuste do ambiente coevolutivo foi feito exclusivamente em cima desse modelo. Somente após esse ajuste, aplicamos o ambiente coevolutivo nos outros três modelos. Assim, “especulamos” que resultados melhores poderiam ser obtidos com a coevolução nos outros três modelos se ajustes específicos fossem realizados, mas que demandariam um tempo de processamento extremamente elevado.

6.2.2 Coevolução versus Evolução Simples

Realizada a comparação entre as abordagens Evolução Conjunta e AGC, onde percebemos uma melhor capacidade de generalização das regras descobertas com o AGC, resolvemos também aplicar as regras descobertas com a Evolução Simples, para cada modelo, nas 50 variações do Gauss18. As regras utilizadas nestes experimentos foram as apresentadas nos experimentos da Seção 5.6. As médias gerais obtidas estão apresentadas na Tabela 6.8, onde também replicamos os resultados com a coevolução. Vemos que em três dos quatro modelos a abordagem coevolutiva foi realmente superior à Evolução Simples. Entretanto, observando os resultados, para nossa surpresa, identificamos que os resultados obtidos na Evolução Simples do Gauss18, Vizinhança Totalística e atributo nível dinâmico, foram um pouco melhores que o melhor resultado com a abordagem do AGC, onde foi utilizado o

modelo de vizinhança selecionada e três atributos diferentes.

Assim, para averiguarmos melhor a qualidade das regras descobertas com as duas abordagens, foram geradas 30 novas variações aleatórias do Gauss18 com um número maior de modificações em relação ao grafo original. Nesse novo conjunto, 10 delas (V18_60 a V18_69) possuem 6 modificações em relação ao grafo original, outras 10 (V18_70 a V18_79) possuem 7 modificações, e as demais (V18_80 a V18_89) possuem 8 modificações. Primeiramente, aplicamos nessas 30 variações as regras descobertas com a Evolução Simples, Vizinhança Totalística e atributo nível dinâmico. Para este experimento, as médias gerais, nas 30 variações, são $T_{med_geral} = 48,93$, $T_{dp_geral} = 3,58$, $T_{min_geral} = 45,16$ e $T_{max_geral} = 58,53$. As tabelas detalhadas são apresentadas no Apêndice D. No experimento seguinte, aplicamos as regras descobertas com a utilização da abordagem coevolutiva, Vizinhança Selecionada e os atributos co-nível estático, custo computacional e custo de comunicação, no escalonamento das 30 novas variações. Calculando-se as médias gerais tem-se que $T_{med_geral} = 48,83$, $T_{dp_geral} = 2,88$, $T_{min_geral} = 46,10$ e $T_{max_geral} = 59,70$. O Apêndice D apresenta a tabela com esse experimento. Comparando-se estes resultados com os obtidos no experimento anterior, percebe-se a similaridade dos mesmos, sendo a média geral um pouco melhor com a utilização do AGC, e o mínimo geral foi um pouco melhor com a Evolução Simples.

Modelo	Evolução Simples	AGC
Vizinhança Selecionada 3 atributos	$T_{med_geral} = 51,4$ $T_{dp_geral} = 3,0$ $T_{min_geral} = 48,8$ $T_{max_geral} = 60,6$	$T_{med_geral} = 48,6$ $T_{dp_geral} = 3,5$ $T_{min_geral} = 45,8$ $T_{max_geral} = 62,8$
Vizinhança Selecionada Atributo nível dinâmico	$T_{med_geral} = 55,0$ $T_{dp_geral} = 6,1$ $T_{min_geral} = 47,5$ $T_{max_geral} = 70,7$	$T_{med_geral} = 49,3$ $T_{dp_geral} = 2,9$ $T_{min_geral} = 46,6$ $T_{max_geral} = 60,0$
Vizinhança Totalística 3 atributos	$T_{med_geral} = 52,0$ $T_{dp_geral} = 3,5$ $T_{min_geral} = 47,7$ $T_{max_geral} = 63,9$	$T_{med_geral} = 51,1$ $T_{dp_geral} = 2,6$ $T_{min_geral} = 46,3$ $T_{max_geral} = 58,0$
Vizinhança Totalística Atributo nível dinâmico	$T_{med_geral} = 48,2$ $T_{dp_geral} = 2,6$ $T_{min_geral} = 45,8$ $T_{max_geral} = 57,5$	$T_{med_geral} = 49,3$ $T_{dp_geral} = 2,8$ $T_{min_geral} = 46,4$ $T_{max_geral} = 56,5$

Tab. 6.8: Aplicação das regras descobertas com a Evolução Simples e AGC no escalonamento das 50 variações aleatórias do Gauss18.

Outra informação importante para a comparação das duas abordagens diz respeito ao tempo de processamento do algoritmo, nas fases de aprendizagem e de operação. Para o experimento com a Evolução Simples, Vizinhança Totalística e atributo nível dinâmico, o tempo gasto na fase de apren-

dizagem, em uma execução do algoritmo, foi de aproximadamente 0,80 hora, enquanto no experimento com o AGC, Vizinhaça Seleccionada e três atributos diferentes o tempo foi de aproximadamente 0,63 hora. Na fase de operação, o tempo de processamento para o experimento com a Vizinhaça Totalística foi de aproximadamente 0,75 hora, e com a Vizinhaça Seleccionada foi 0,26 hora. Assim, como conclusão final dessa comparação evolução simples versus coevolução, constatamos que os melhores experimentos nas cinquenta variações foram os que utilizaram a Evolução Simples na Vizinhaça Totalística e atributo nível dinâmico e a coevolução na Vizinhaça Seleccionada com três atributos. Consideramos as capacidades de generalização das regras evoluídas nesses dois experimentos extremamente similares. Essa similaridade foi constatada também nas 30 variações adicionais. Entretanto, o experimento com a coevolução e a Vizinhaça Seleccionada mostrou-se mais rápido que o experimento com a evolução simples e a Vizinhaça Totalística, especialmente na fase de operação, onde é cerca de três vezes mais rápido.

A Tabela 6.9 apresenta os tempos médios de processamento, em horas, de execução médios envolvidos em todos os experimentos da Seção 6.2. Os tempos de processamento na fase de operação são os tempos médios da aplicação das regras para as cinquenta variações do Gauss18.

Modelo Vizinhaça	Fase de aprendizagem			Fase de operação		
	Simples	Conjunta	Coevolução	Simples	Conjunta	Coevolução
Seleccionada e 3 atributos	0,20	0,52	0,63	0,26	0,26	0,26
Seleccionada e nível dinâmico	0,50	1,33	3,00	1,28	1,28	1,28
Totalística e 3 atributos	0,17	0,37	0,34	0,067	0,067	0,067
Totalística e nível dinâmico	0,80	1,33	2,00	0,75	0,75	0,75

Tab. 6.9: Tempos, em horas, para execução do algoritmo de escalonamento baseado em ACs para as abordagens Simples, Conjunta e Coevolutiva.

6.3 Resultados com os modelos de vizinhaça não-lineares e modo de operação paralelo

Poucos experimentos foram apresentados na literatura, principalmente com a utilização do grafo de programa Gauss18, onde foi utilizado o modo de operação paralelo de AC. Resultados publicados mostraram que, para o Gauss18, o melhor modo de operação de AC utilizado foi o sequencial. Por este motivo, em seções anteriores, concentramos nossos esforços na realização de experimentos que

utilizaram este modo de operação. Porém, devido à importância da natureza paralela dos autômatos celulares, resolvemos realizar alguns experimentos finais e explorar também o modo de operação paralelo dos ACs. Nesta seção, são apresentados resultados experimentais obtidos com a utilização dos modelos de vizinhança não-lineares e com o modo de operação de ACs paralelo (síncrono).

Em [98, 99], uma nova característica foi adicionada para o escalonador baseado em ACs, onde um algoritmo genético coevolutivo (AGC) (Seção 2.10) foi utilizado na descoberta de regras de transição de ACs, no modo de operação paralelo, aptas a escalonar um determinado grafo de programa. Este AGC também é baseado no paradigma *presa-predador*, evoluindo duas populações, uma delas composta por regras de transição e a outra composta por diferentes alocações das tarefas de um grafo de programa em um sistema com dois processadores. No entanto, nos experimentos publicados não foram apresentados resultados para o grafo de programa Gauss18, considerado o mais difícil e o principal grafo do nosso trabalho. Além disso, a abordagem de AGC utilizada por Seredynski e colegas é diferente da abordagem que utilizamos, apresentada na Seção 6.2. Em primeiro lugar, o objetivo do uso da estratégia coevolutiva em [98] e [99] é melhorar a busca evolutiva por regras voltadas à solução de um único grafo, sem a preocupação com a capacidade de generalização. Por isso, a segunda população em [98] e [99] é formada por configurações iniciais, enquanto que em nossa abordagem ela é formada por grafos de programa.

6.3.1 Evolução Simples

Resultados experimentais utilizando o Gauss18, o AG Padrão na fase de aprendizagem e o modo de operação paralelo de AC não foram detalhados na literatura. Por isso, resolvemos realizar alguns experimentos com o modo paralelo. Os parâmetros do AC utilizados nos experimentos foram $CI_s = 25$ e $M = 50$. Para o AG utilizado (Seção 5.1.4), os parâmetros foram $P = 100$, $E = 10$, $p_c = 0.90$, $G = 100$, $p_m = 0.012$ para a *Vizinhança Seleccionada* e $p_m = 0.024$ para a *Vizinhança Totalística*. A política de escalonamento utilizada foi “a tarefa com o maior nível dinâmico é executada primeiro”.

No primeiro experimento, utilizamos a vizinhança seleccionada e, para a formação da vizinhança, foi utilizado o atributo co-nível estático para o conjunto de tarefas *predecessoras(k)*, o custo computacional para as *irmãs(k)* e custo de comunicação para as *sucessoras(k)*. A Figura 6.6a apresenta a evolução do melhor indivíduo (regra) da população na fase de aprendizagem do algoritmo de esca-

lonamento. Na primeira geração, o valor médio de T para o melhor indivíduo, calculado para as 25 CIs usadas como testes de problema, é igual a 61,88. Durante as próximas gerações, o AG modifica as regras até que, na geração 49, é descoberta uma regra capaz de evoluir, por 50 passos de tempo, as 25 configurações iniciais para configurações finais que representam o escalonamento ótimo para o Gauss18. Entretanto, essa regra não passa em um teste de problema criado na geração 55. O AG rapidamente modifica as regras, descobrindo assim, na geração 56, uma regra que passa em todos os testes subsequentes.

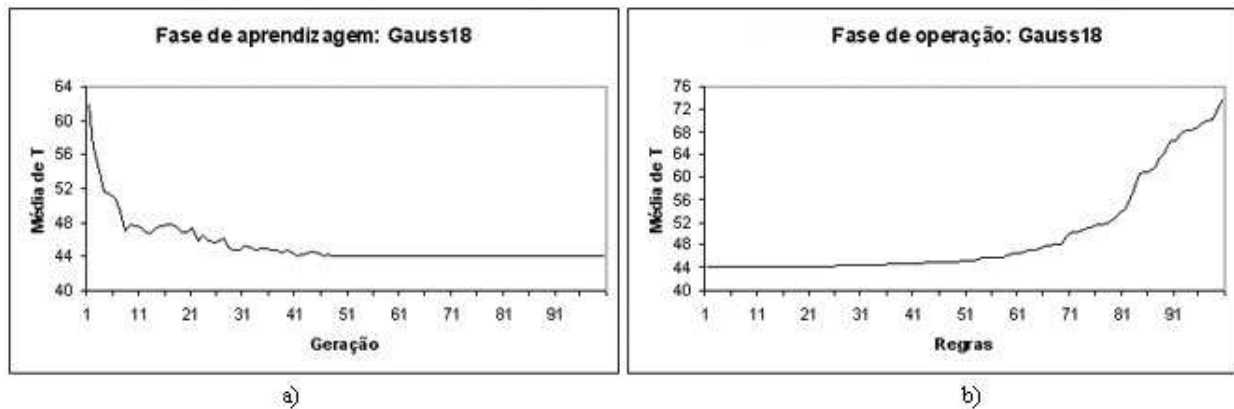


Fig. 6.6: (a) Fase de aprendizagem do algoritmo de escalonamento para o Gauss18: Vizinhança Seleccionada, modo paralelo de AC e três atributos diferentes para a formação da vizinhança. (b) Fase de operação.

Na fase de operação, 100 novas configurações iniciais aleatórias foram usadas para testar as regras evoluídas. Das 100 regras, 15 foram capazes de evoluir as 100 configurações iniciais para configurações finais que representam o escalonamento ótimo para o Gauss18 ($T = 44,0$). A Figura 6.6b apresenta a fase de operação para esse experimento. Comparando-se este resultado com o experimento similar descrito na Seção 5.6.1, onde o modo de operação de AC utilizado foi o sequencial, percebe-se que os resultados obtidos com o modo sequencial foram melhores, visto que cerca de 50 das 100 regras evoluídas conseguiram escalonar perfeitamente o Gauss18 em 100% dos testes.

O segundo experimento foi similar ao primeiro, exceto pelo fato de que utilizamos o atributo nível dinâmico na formação da vizinhança. A melhor regra evoluída apresentou o valor médio de T igual a 44,36 na última geração. Assim, a melhor regra descoberta pelo AG na fase de aprendizagem não foi capaz de convergir todas as 25 CIs, na última geração, para alocações que representam o escalonamento ótimo para o Gauss18. Logo, utilizando o atributo nível dinâmico na vizinhança

selecionada e o modo de operação paralelo de AC, foi possível encontrar regras que levam apenas a soluções sub-ótimas para o problema do escalonamento para o Gauss18, e não foram realizados testes na fase de operação.

Em outro experimento, utilizamos a *Vizinhança Totalística* e os atributos co-nível estático, custo computacional e custo de comunicação para a formação da vizinhança das tarefas. O AG não conseguiu descobrir nenhuma regra de transição capaz de convergir as 25 CIs para alocações ótimas. O valor de aptidão médio da melhor regra foi igual a 51,76, na última geração, valor este distante do T médio ótimo ($T = 44,0$). Portanto, também não foi realizada a fase de operação para esse modelo.

O último experimento realizado com o AG padrão foi similar ao experimento anterior, exceto pelo fato de que utilizamos apenas o atributo nível dinâmico para a formação da vizinhança das tarefas do Gauss18. Assim como no exemplo anterior, os resultados encontrados não foram de boa qualidade onde, na última geração, o valor médio de T para a melhor regra da população foi igual a 51,12.

Conforme podemos perceber através dos resultados experimentais aqui apresentados, a *Vizinhança Selecionada* mostrou ser, assim como em outros experimentos, o melhor modelo de vizinhança para o Gauss18, principalmente com a utilização dos atributos co-nível estático, custo computacional e custo de comunicação na formação da vizinhança. Além disso, comparando-se também os resultados obtidos para os modos de operação seqüencial e paralelo de AC, o modo seqüencial apresentou melhores resultados, conforme já havíamos observado no modelo linear (Seção 5.5).

6.3.2 Evolução Conjunta

Apresentamos a seguir os experimentos realizados com a Evolução Conjunta utilizando-se os modelos de vizinhança selecionada e totalística em ACs com modo de operação paralelo. Os parâmetros para o AG e o AC utilizados foram $P = 100$, $E = 10$, $p_c = 0,90$, $p_m = 0,012$, $G = 100$, $CIs = 25$ e $M = 50$ e a política de escalonamento utilizada “a tarefa com o maior nível dinâmico é executada primeiro”.

O primeiro experimento que executamos utilizou o modelo de vizinhança selecionada, onde foram utilizados três atributos diferentes para a formação da vizinhança - o co-nível estático para as *predecessoras(k)*, o custo computacional para as *irmãs(k)* e o custo de comunicação para as *sucessoras(k)*. Primeiramente, as regras descobertas com a abordagem Evolução Simples, foram utilizadas no escalonamento das quinze variações do Gauss18. Foram gerados 100 testes de problema para cada grafo

de programa, e estes foram utilizados para avaliar cada regra da população. Os resultados obtidos estão apresentados na Tabela 6.10 (Evolução Simples), onde os resultados dos experimentos de referência são repetidos para facilitar a compreensão. Como podemos observar, nenhuma regra foi apta a encontrar o escalonamento ótimo em nenhuma das variações. Para os grafos Gauss18_6, Gauss18_7, Gauss18_9, Gauss18_10, Gauss18_11 e Gauss18_12, os valores médios (T_{med}) de T aproximaram-se dos valores obtidos pelo AG de referência (T_{AG_Ref}). Calculando-se o intervalo de confiança para os valores obtidos, estamos 95% confiantes de que a média do tempo total de execução T , nas 15 variações, está no intervalo [52,61; 54,37]. Nas 10 variações não utilizadas na fase de aprendizagem, este intervalo é [50,62; 52,38].

Grafo	Experimentos de Referência		Evolução Simples				Evolução Conjunta			
	Alocações Aleatórias	T_{AG_Ref}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{med}	T_{max}
Gauss18_1	75,92	47,0	62,21	4,97	51,0	70,0	49,0	0,00	49,0	49,0
Gauss18_2	77,38	47,0	64,69	5,51	52,0	74,0	52,18	1,26	52,0	61,0
Gauss18_3	70,67	47,0	52,37	3,49	49,0	71,0	49,0	0,00	49,0	49,0
Gauss18_4	75,19	47,0	52,45	3,51	50,0	71,0	49,0	0,00	49,0	49,0
Gauss18_5	77,71	50,0	55,65	5,06	52,0	87,0	51,00	0,00	51,0	51,0
Gauss18_6	74,0	47,0	48,0	0,00	48,0	48,0	53,19	1,45	53,0	67,0
Gauss18_7	78,0	44,0	44,14	1,40	44,0	58,0	47,0	0,00	47,0	47,0
Gauss18_8	79,0	44,0	59,02	5,25	47,0	69,0	47,18	1,27	47,0	57,0
Gauss18_9	79,0	47,0	48,0	0,00	48,0	48,0	49,05	0,50	49,0	54,0
Gauss18_10	80,0	44,0	47,81	9,84	44,0	83,0	49,49	2,29	49,0	66,0
Gauss18_11	78,08	47,0	48,66	6,95	47,0	93,0	49,20	1,51	49,0	63,0
Gauss18_12	74,10	47,0	48,14	1,40	48,0	62,0	51,00	0,00	51,0	51,0
Gauss18_13	71,22	44,0	58,67	4,99	47,0	68,0	45,00	0,00	45,0	45,0
Gauss18_14	77,98	46,0	60,89	5,62	48,0	70,0	48,18	1,26	48,0	57,0
Gauss18_15	74,13	47,0	51,69	9,37	48,0	86,0	49,24	1,37	49,0	57,0
Média (10)	76,55	45,70	51,50	4,48	46,90	68,50	48,85	0,96	48,70	56,40
Média (15)	76,15	46,33	53,49	4,49	48,20	70,53	49,24	0,72	49,13	54,86

Tab. 6.10: Resultados obtidos com a Evolução Simples do Gauss18 e com a Evolução Conjunta do Gauss18 e 5 variações deste grafo de programa, utilizando-se o modo de operação paralelo, Vizinhança Seleccionada e atributos co-nível estático, custo computacional e custo de comunicação.

Realizamos o experimento com a abordagem Evolução Conjunta. Em trinta execuções do AG foi obtida uma média de T para o melhor indivíduo na última geração de 299,0, na melhor execução, ilustrada na Figura 6.7. Na primeira geração, a média de T para o melhor indivíduo é 396,52. O AG modifica as regras e, na geração 100, o valor 299,0 é alcançado.

Na fase de operação, as regras descobertas foram utilizadas no escalonamento das quinze variações do Gauss18. Foram gerados 100 testes de problema para cada variação, sendo estes utilizados na avaliação de cada regra. Os resultados obtidos estão apresentados na Tabela 6.10 (Evolução Con-

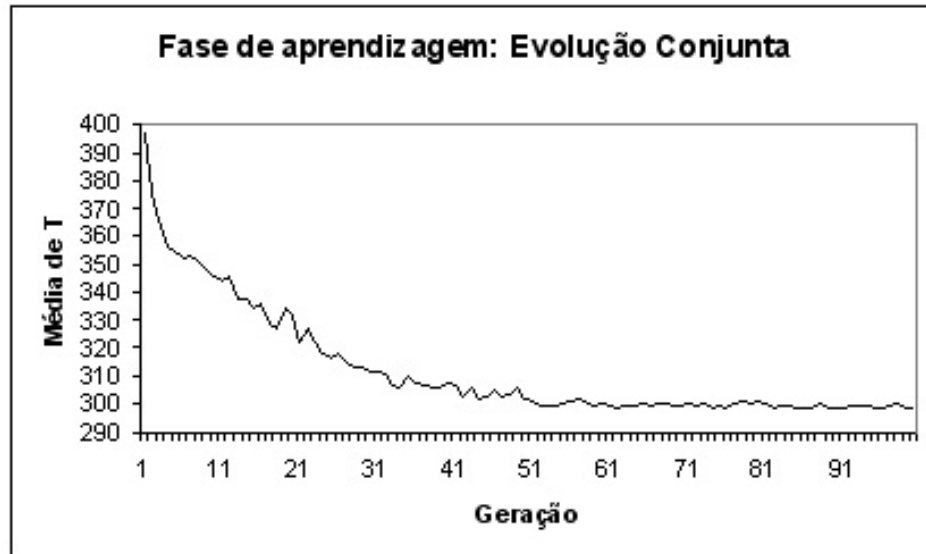


Fig. 6.7: Fase de aprendizagem do algoritmo de escalonamento: modo de operação paralelo, Evolução Conjunta, Vizinhança Seleccionada com três atributos diferentes.

junta). O valor médio de T_{med} foi menor na Evolução Conjunta, tanto considerando-se as dez variações quanto considerando-se as quinze. Considerando-se as quinze variações, a Evolução Conjunta apresentou um melhor tempo médio em nove delas. Já nas dez variações que não participaram da fase de aprendizagem, a Evolução Simples foi melhor em seis. No resultado médio, a Evolução Conjunta se mostrou superior também nas dez variações. Calculando-se o intervalo de confiança para este experimento, estamos 95% confiantes de que o valor médio de T está no intervalo $[49,10; 49,39]$ para as quinze variações e $[48,66; 49,04]$ nas outras dez. Também avaliamos o teste de hipótese nula para comparar os dois experimentos. Há evidências de que os tempos de execução obtidos pelas regras descobertas no experimento da evolução conjunta são melhores que os obtidos no experimento da evolução simples. Estamos 95% confiantes de que esta melhoria está entre 3,35 e 5,13 unidades de tempo nas quinze variações, e entre 1,75 e 3,54 unidades de tempo, considerando-se apenas as dez variações não participantes da fase de aprendizagem. Estes resultados qualificam as regras descobertas pela evolução conjunta como sendo mais gerais.

O tempo de processamento médio desses experimentos, na fase de aprendizagem, é 0,067 hora para a Evolução Simples e 0,44 hora para a Evolução Conjunta. Na fase de operação, em ambas as abordagens, o tempo médio de processamento do algoritmo é de 0,042 hora.

Em sequência, foram realizados os experimentos comparativos entre a Evolução Simples e a

Evolução Conjunta, considerando-se as quinze variações da Seção 5.3, nos outros três modelos de vizinhança/atributos investigados nessa dissertação. São eles: Vizinhança Seleccionada e atributo nível dinâmico, Vizinhança Totalística e três atributos, e Vizinhança Totalística e atributo nível dinâmico. Os resultados obtidos nesses experimentos são apresentados de forma detalhada no Apêndice D e são resumidos na Tabela 6.11. Conforme podemos observar, para os quatro modelos de vizinhança/atributos, a Evolução Conjunta se mostrou melhor que a Evolução Simples, com relação ao valor de T_{med_geral} . Esta melhoria foi mais significativa com o modelo de vizinhança seleccionada. Vale ressaltar que, tanto na literatura quanto nos experimentos que realizamos e apresentamos nessa dissertação (Seções B.2, 5.6.1, 6.1, 6.2 e 6.3.1), o modelo de vizinhança que obteve melhores resultados para o Gauss18 foi a Vizinhança Seleccionada, o que também ficou evidente nesses experimentos com o modo de operação paralelo.

Modelo	Evolução Simples		Evolução Conjunta	
	10 variações	15 variações	10 variações	15 variações
Vizinhança Seleccionada e 3 atributos	$T_{med_geral} = 51,50$ $T_{dp_geral} = 4,48$ $T_{min_geral} = 46,90$ $T_{max_geral} = 68,50$	$T_{med_geral} = 53,49$ $T_{dp_geral} = 4,49$ $T_{min_geral} = 48,20$ $T_{max_geral} = 70,53$	$T_{med_geral} = 48,85$ $T_{dp_geral} = 0,96$ $T_{min_geral} = 48,70$ $T_{max_geral} = 56,40$	$T_{med_geral} = 49,24$ $T_{dp_geral} = 0,72$ $T_{min_geral} = 49,13$ $T_{max_geral} = 54,86$
Vizinhança Seleccionada e atributo nível dinâmico	$T_{med_geral} = 55,15$ $T_{dp_geral} = 7,01$ $T_{min_geral} = 50,10$ $T_{max_geral} = 86,00$	$T_{med_geral} = 56,17$ $T_{dp_geral} = 7,62$ $T_{min_geral} = 51,13$ $T_{max_geral} = 87,26$	$T_{med_geral} = 47,25$ $T_{dp_geral} = 0,88$ $T_{min_geral} = 47,00$ $T_{max_geral} = 50,60$	$T_{med_geral} = 47,79$ $T_{dp_geral} = 0,92$ $T_{min_geral} = 47,20$ $T_{max_geral} = 51,33$
Vizinhança Totalística e 3 atributos	$T_{med_geral} = 53,56$ $T_{dp_geral} = 3,57$ $T_{min_geral} = 49,0$ $T_{max_geral} = 61,60$	$T_{med_geral} = 54,75$ $T_{dp_geral} = 3,58$ $T_{min_geral} = 50,13$ $T_{max_geral} = 62,80$	$T_{med_geral} = 52,24$ $T_{dp_geral} = 1,24$ $T_{min_geral} = 49,20$ $T_{max_geral} = 57,80$	$T_{med_geral} = 52,53$ $T_{dp_geral} = 1,20$ $T_{min_geral} = 50,26$ $T_{max_geral} = 57,80$
Vizinhança Totalística e atributo nível dinâmico	$T_{med_geral} = 54,67$ $T_{dp_geral} = 3,19$ $T_{min_geral} = 51,50$ $T_{max_geral} = 70,0$	$T_{med_geral} = 56,24$ $T_{dp_geral} = 4,48$ $T_{min_geral} = 51,86$ $T_{max_geral} = 74,73$	$T_{med_geral} = 51,23$ $T_{dp_geral} = 1,93$ $T_{min_geral} = 47,50$ $T_{max_geral} = 59,10$	$T_{med_geral} = 52,24$ $T_{dp_geral} = 1,60$ $T_{min_geral} = 49,13$ $T_{max_geral} = 58,66$

Tab. 6.11: Resultados resumidos da comparação Evolução Simples versus Evolução Conjunta no modo de operação paralelo de AC.

Um resultado também importante e que nos chamou atenção foi a aplicação das regras descobertas com a Evolução Simples e com a Evolução Conjunta para escalonar o próprio Gauss18 no modelo com a Vizinhança Seleccionada e o atributo nível dinâmico. A melhor regra descoberta com a Evolução Simples, quando aplicada em 100 testes de problema gerados para o Gauss18 original, apresentou os seguintes tempos: $T_{med} = 55,88$, $T_{dp} = 8,04$, $T_{min} = 44,0$ e $T_{max} = 88,0$. A melhor regra descoberta com a Evolução Conjunta obteve $T_{med} = 47,28$, $T_{dp} = 1,02$, $T_{min} = 47,0$ e $T_{max} = 51,0$. Nós entendemos que a “dificuldade” de adaptabilidade das regras imposta pela Evolução Con-

junta no algoritmo de escalonamento, contribuiu para a descoberta de regras com um maior poder de escalonamento para o próprio Gauss18. Mesmo que na Evolução Simples apenas o Gauss18 seja utilizado em todas gerações, o AG não conseguiu encontrar regras de transição de AC capazes de encontrar o escalonamento ótimo ou até mesmo sub-ótimo para este grafo. Provocada uma “perturbação” na fase de aprendizagem, o AG foi capaz de descobrir indivíduos melhores, aptos a encontrar um escalonamento sub-ótimo inclusive para o Gauss18.

6.3.3 Coevolução

Nesta seção, apresentamos experimentos onde utilizamos o modo de operação paralelo de AC e a abordagem de AGC para a fase de aprendizagem do algoritmo de escalonamento. Nestes experimentos, os parâmetros do AC e AG utilizados foram $P = 100$, $E = 10$, $p_c = 0,90$, $p_m = 0,012$, $G = 100$, $CIs = 25$ e $M = 50$. A política de escalonamento utilizada foi “a tarefa com o maior nível dinâmico é executada primeiro”.

Nas Seções 5.6.1 e 6.2, apresentamos experimentos com o modo de operação seqüencial de AC, onde os melhores resultados obtidos foram com a utilização da *Vizinhança Seleccionada* e os atributos co-nível estático, custo computacional e custo de comunicação para a formação da vizinhança. Assim, optamos por iniciar os experimentos com a abordagem coevolutiva e o modo de operação paralelo também com esse modelo de vizinhança.

Inicialmente, utilizamos as regras descobertas em um experimento descrito anteriormente, onde foi utilizado apenas o Gauss18 (Evolução Simples) na fase de aprendizagem do algoritmo de escalonamento, para escalonar as cinquenta variações do Gauss18 geradas aleatoriamente e apresentadas na Seção 6.2. Os resultados obtidos estão listados na Tabela 6.12. Os valores médios gerais, nas 50 variações, são $T_{med_geral} = 57,12$, $T_{dp_geral} = 7,78$, $T_{min_geral} = 48,50$ e $T_{max_geral} = 81,30$. Calculamos o intervalo de confiança para as regras de AC para este experimento e estamos 95% confiantes que a média do tempo total de execução T , nas cinquenta variações do Gauss18, está no intervalo [55,59; 58,64].

No segundo experimento aplicamos, nas 50 variações do Gauss18, as regras descobertas através da abordagem Evolução Conjunta, *Vizinhança Seleccionada* e os três atributos diferentes, descrito na Seção 6.3.2. Os resultados são apresentados na Tabela 6.13. Os valores médios gerais, nas 50 varia-

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	53,61	13,53	44,0	87,0	V18_20	46,8	6,57	44,0	69,0
V18_11	64,62	9,01	50,0	78,0	V18_21	57,8	3,68	53,0	69,0
V18_12	60,0	8,91	53,0	85,0	V18_22	57,16	8,65	46,0	97,0
V18_13	49,19	10,32	44,0	95,0	V18_23	58,57	7,53	49,0	84,0
V18_14	44,0	0,00	44,0	44,0	V18_24	68,39	8,76	51,0	94,0
V18_15	57,14	10,94	47,0	95,0	V18_25	60,31	10,17	53,0	95,0
V18_16	61,07	9,27	53,0	94,0	V18_26	59,65	8,75	48,0	85,0
V18_17	47,81	6,09	46,0	82,0	V18_27	44,0	0,00	44,0	44,0
V18_18	56,92	13,87	44,0	102,0	V18_28	44,69	3,67	44,0	75,0
V18_19	59,89	1,75	53,0	63,0	V18_29	53,33	4,74	44,0	73,0
Média	55,42	8,36	47,80	82,50	Média	55,07	6,25	47,60	78,50
V18_30	51,13	3,81	44,0	69,0	V18_40	59,55	9,66	46,0	75,0
V18_31	47,81	6,09	46,0	82,0	V18_41	65,1	8,60	55,0	85,0
V18_32	67,05	12,36	52,0	89,0	V18_42	53,78	3,00	47,0	65,0
V18_33	51,32	10,35	44,0	79,0	V18_43	59,91	9,09	52,0	76,0
V18_34	50,0	0,00	50,0	50,0	V18_44	60,78	7,40	46,0	78,0
V18_35	48,64	6,64	47,0	83,0	V18_45	64,23	14,22	50,0	95,0
V18_36	64,31	1,63	64,0	76,0	V18_46	63,61	8,71	44,0	90,0
V18_37	53,82	6,91	44,0	82,0	V18_47	57,35	11,17	50,0	84,0
V18_38	60,76	9,08	46,0	82,0	V18_48	60,23	4,19	56,0	79,0
V18_39	56,11	14,73	44,0	92,0	V18_49	64,64	7,01	52,0	87,0
Média	55,09	7,16	48,10	78,40	Média	60,91	8,30	49,80	81,40
V18_50	68,46	7,22	59,0	94,0	Média geral	57,12	7,78	48,50	81,30
V18_51	50,23	11,40	46,0	93,0					
V18_52	53,79	2,68	53,0	72,0					
V18_53	72,04	14,77	48,0	94,0					
V18_54	61,24	9,61	51,0	85,0					
V18_55	54,94	12,51	45,0	88,0					
V18_56	63,38	5,46	52,0	77,0					
V18_57	49,47	1,78	49,0	64,0					
V18_58	50,94	10,17	44,0	82,0					
V18_59	66,46	12,96	45,0	108,0					
Média	59,09	8,85	49,20	85,70					

Tab. 6.12: Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a Evolução Simples do Gauss18, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.

ções, são $T_{med_geral} = 56,27$, $T_{dp_geral} = 6,23$, $T_{min_geral} = 50,88$ e $T_{max_geral} = 79,88$. Calculamos o intervalo de confiança para este experimento e estamos 95% confiantes que a média do tempo total de execução T , nas cinquenta variações do Gauss18, está no intervalo $[55,05; 57,49]$. Avaliamos o teste de hipótese nula para comparar os experimentos com a Evolução Simples e Conjunta. Não existem fortes evidências de que os tempos de execução obtidos pelas regras descobertas no experimento da evolução conjunta são melhores que os tempos obtidos no experimento da evolução simples. Estamos 95% confiantes de que o ganho/perda com a evolução conjunta está entre -1,21 e 2,90 unidades de tempo. Comparando-se o valor T_{med_geral} para os dois experimentos, observamos que, neste caso, a Evolução Conjunta (56,27) foi um pouco superior à Evolução Simples (57,12).

Em outro experimento, utilizamos a abordagem coevolutiva na fase de aprendizagem do algoritmo

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,0	0,00	49,0	49,0	V18_20	57,9	3,18	53,0	73,0
V18_11	53,29	2,05	53,0	69,0	V18_21	61,63	9,37	55,0	84,0
V18_12	57,34	9,60	52,0	94,0	V18_22	49,68	3,46	49,0	73,0
V18_13	49,86	3,08	49,0	67,0	V18_23	51,0	0,00	51,0	51,0
V18_14	49,59	2,43	49,0	63,0	V18_24	51,44	3,30	51,0	83,0
V18_15	50,19	2,34	49,0	63,0	V18_25	61,24	9,47	52,0	83,0
V18_16	50,1	4,95	49,0	83,0	V18_26	51,02	7,07	49,0	85,0
V18_17	51,97	5,55	49,0	82,0	V18_27	49,59	2,43	49,0	63,0
V18_18	53,29	6,72	49,0	73,0	V18_28	57,72	7,83	52,0	85,0
V18_19	49,49	1,78	49,0	57,0	V18_29	57,53	11,74	49,0	94,0
Média	51,41	3,85	49,70	70,0	Média	54,87	5,78	51,0	77,4
V18_30	53,0	8,45	49,0	77,0	V18_40	53,03	8,29	49,0	78,0
V18_31	53,11	5,11	49,0	82,0	V18_41	64,31	8,00	57,0	83,0
V18_32	67,3	11,49	55,0	110,0	V18_42	55,9	4,79	51,0	68,0
V18_33	49,82	3,98	45,0	79,0	V18_43	57,86	3,35	56,0	72,0
V18_34	56,79	8,38	49,0	85,0	V18_44	59,03	4,56	55,0	84,0
V18_35	46,42	2,24	46,0	64,0	V18_45	63,74	6,45	47,0	83,0
V18_36	54,22	2,20	54,0	76,0	V18_46	62,17	8,32	49,0	78,0
V18_37	51,39	8,58	49,0	102,0	V18_47	55,63	10,21	49,0	86,0
V18_38	47,53	1,89	47,0	62,0	V18_48	63,96	4,71	62,0	100,0
V18_39	53,76	9,71	49,0	85,0	V18_49	67,67	9,30	54,0	91,0
Média	53,33	6,20	49,20	82,20	Média	60,33	6,79	52,90	82,23
V18_50	62,05	8,07	51,0	80,0	Média geral	56,27	6,23	50,88	79,88
V18_51	66,32	10,06	52,0	95,0					
V18_52	55,44	4,99	53,0	77,0					
V18_53	72,75	14,19	52,0	101,0					
V18_54	66,85	10,67	51,0	92,0					
V18_55	58,91	10,37	48,0	88,0					
V18_56	57,91	9,79	51,0	94,0					
V18_57	55,47	6,82	52,0	89,0					
V18_58	53,88	5,22	51,0	71,0					
V18_59	64,64	5,16	55,0	89,0					
Média	61,42	8,53	51,60	87,60					

Tab. 6.13: Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.

de escalonamento. A população *Pop2* inicial do AGC foi composta pelo Gauss18 e mais nove grafos de programa gerados aleatoriamente, da mesma forma utilizada nos experimentos com o modo de operação sequencial descritos na Seção 6.2.

Realizamos trinta execuções do algoritmo de escalonamento. A população de regras evoluídas foi avaliada aplicando-a nas cinquenta variações do Gauss18. Foram gerados 100 testes de problema para cada variação, sendo estes testes utilizados para avaliar cada regra da população. Os resultados obtidos para a melhor das trinta execuções estão listados na Tabela 6.14. Os valores médios gerais, nas 50 variações, são $T_{med_geral} = 49,71$, $T_{dp_geral} = 1,78$, $T_{min_geral} = 48,30$ e $T_{max_geral} = 56,60$. A Figura 6.8 apresenta o valor de aptidão médio, em cada geração do AG, do melhor indivíduo das populações de regras (*Pop1*) e de grafos de programa (*Pop2*). Na primeira geração do AG, a aptidão

média do melhor indivíduo de *Pop1* é 65,29 e 79,46 em *Pop2*. Em *Pop1*, a aptidão média do melhor indivíduo tende a diminuir com o passar das gerações, uma vez que as melhores regras (menor valor médio de T) são escolhidas pelo AG para permanecerem e gerar filhos, alcançando o valor 51,01 na última geração. Em *Pop2*, a partir da geração 30, a aptidão média do melhor indivíduo oscila entre 45 e 70 até que, na última geração, este valor é 50,11.

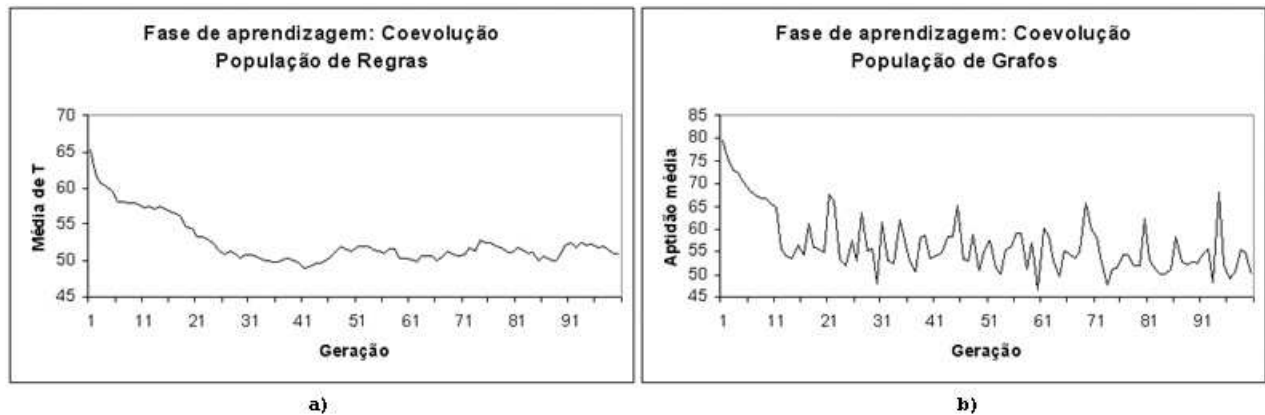


Fig. 6.8: Fase de aprendizagem do algoritmo de escalonamento baseado em ACs com modo de operação paralelo: processo coevolutivo das populações de regras e grafos de programa.

Calculamos o intervalo de confiança para as regras evoluídas neste experimento e estamos 95% confiantes que a média do tempo total de execução T , nas cinquenta variações do Gauss18, está no intervalo [49,36; 50,06]. Também avaliamos o teste de hipótese nula para comparar os resultados obtidos com a Evolução Conjunta e com a abordagem coevolutiva. Há evidências significativas de que o tempo de execução obtido pelas regras co-evoluídas é melhor que o obtido no experimento da evolução conjunta. Estamos 95% confiantes de que esta melhoria está entre 5,22 e 7,88 unidades de tempo. Aplicando-se o teste de hipótese nula para comparar os experimentos com a Evolução Simples e a abordagem coevolutiva, estamos 95% confiantes de que a melhoria com o emprego da coevolução está entre 5,79 e 9,00 unidades de tempo. Portanto, as regras descobertas pela abordagem coevolutiva são mais gerais que aquelas obtidas tanto pela evolução conjunta quanto pela evolução simples.

Posteriormente, realizamos experimentos semelhantes aos descritos acima para os outros modelos de vizinhança - a *Vizinhança Seleccionada* com o atributo nível dinâmico, a *Vizinhança Totalística* com os três atributos diferentes e também apenas com o atributo nível dinâmico. Os resultados obtidos para cada modelo e abordagem estão resumidos na Tabela 6.15. Os resultados detalhados encontram-

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	47,85	1,88	47,0	52,0	V18_20	47,82	0,98	47,0	49,0
V18_11	51,18	0,57	51,0	53,0	V18_21	53,76	2,49	53,0	78,0
V18_12	47,0	0,0	47,0	47,0	V18_22	47,87	3,35	47,0	67,0
V18_13	47,23	2,30	47,0	70,0	V18_23	49,04	0,28	49,0	51,0
V18_14	47,34	0,75	47,0	49,0	V18_24	52,28	5,46	47,0	61,0
V18_15	47,7	0,95	47,0	49,0	V18_25	47,06	0,34	47,0	49,0
V18_16	47,18	1,80	47,0	65,0	V18_26	47,12	0,47	47,0	49,0
V18_17	46,38	0,48	46,0	47,0	V18_27	47,34	0,75	47,0	49,0
V18_18	47,56	1,99	47,0	65,0	V18_28	47,0	0,0	47,0	47,0
V18_19	47,16	0,54	47,0	49,0	V18_29	52,4	2,34	49,0	54,0
Média	47,65	1,12	47,30	54,60	Média	49,16	1,64	48,0	55,40
V18_30	47,33	1,58	47,0	62,0	V18_40	47,42	0,81	47,0	49,0
V18_31	46,38	0,48	46,0	47,0	V18_41	53,28	0,69	53,0	55,0
V18_32	48,6	0,92	48,0	50,0	V18_42	46,43	0,49	46,0	47,0
V18_33	51,47	5,74	47,0	69,0	V18_43	49,0	0,0	49,0	49,0
V18_34	46,24	0,42	46,0	47,0	V18_44	49,88	3,76	46,0	65,0
V18_35	50,57	3,04	48,0	61,0	V18_45	49,96	1,65	48,0	53,0
V18_36	52,26	0,67	52,0	54,0	V18_46	47,82	0,53	47,0	49,0
V18_37	47,52	1,41	47,0	53,0	V18_47	54,94	5,22	50,0	74,0
V18_38	49,69	2,23	47,0	53,0	V18_48	53,16	0,54	53,0	55,0
V18_39	47,0	0,0	47,0	47,0	V18_49	53,0	0,0	53,0	53,0
Média	48,70	1,64	47,50	54,30	Média	50,48	1,36	49,20	54,90
V18_50	56,66	5,06	53,0	77,0	Média geral	49,71	1,78	48,30	56,60
V18_51	49,24	0,65	49,0	51,0					
V18_52	47,67	4,23	47,0	84,0					
V18_53	53,54	3,29	51,0	68,0					
V18_54	49,0	0,0	49,0	49,0					
V18_55	51,27	4,52	47,0	64,0					
V18_56	50,94	3,07	47,0	54,0					
V18_57	53,48	0,85	53,0	55,0					
V18_58	52,03	2,78	47,0	61,0					
V18_59	61,93	6,97	52,0	75,0					
Média	52,57	3,14	49,50	63,80					

Tab. 6.14: Valores obtidos para 50 variações do Gauss18: aplicação de regras descobertas com a abordagem coevolutiva do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.

se no Apêndice D. Como pode ser observado, com exceção apenas do experimento com a *Vizinhança Totalística* e o atributo nível dinâmico, em todos os experimentos a abordagem coevolutiva apresentou uma média geral (T_{med_geral}) menor que a média geral obtida com a Evolução Conjunta que, por sua vez, apresentou a média geral menor que a média obtida com a Evolução Simples. Além disso, no experimento com a *Vizinhança Seleccionada* e o atributo nível dinâmico, a abordagem coevolutiva mostrou-se um pouco melhor que o experimento detalhado anteriormente, onde utilizamos três atributos diferentes ao invés do atributo nível dinâmico. No entanto, a diferença na média geral dos dois experimentos é pequena (0,02), e o tempo de processamento do algoritmo de escalonamento quando se utiliza os três atributos diferentes é de aproximadamente 0,83 hora, enquanto com o atributo nível dinâmico é de aproximadamente 3 horas.

Finalmente, comparando-se com o modo de operação seqüencial (Seção 6.2), percebemos que a adoção da estratégia coevolutiva retornou melhorias ainda mais significativas nesse modo de operação. Comparando-se as Tabelas 6.8 e 6.15, percebemos que nos experimentos com a Evolução Simples, o modo de operação seqüencial foi expressivamente superior. Entretanto, com a coevolução, os resultados obtidos no modo de operação paralelo se aproximam bastante dos obtidos com o modo seqüencial.

Modelo	Evolução Simples	Evolução Conjunta	AGC
Vizinhança Seleccionada 3 atributos	$T_{med_geral} = 57,11$ $T_{dp_geral} = 7,78$ $T_{min_geral} = 48,50$ $T_{max_geral} = 81,30$	$T_{med_geral} = 56,27$ $T_{dp_geral} = 6,23$ $T_{min_geral} = 50,88$ $T_{max_geral} = 79,88$	$T_{med_geral} = 49,71$ $T_{dp_geral} = 1,78$ $T_{min_geral} = 48,30$ $T_{max_geral} = 56,60$
Vizinhança Seleccionada Nível dinâmico	$T_{med_geral} = 58,94$ $T_{dp_geral} = 8,17$ $T_{min_geral} = 49,12$ $T_{max_geral} = 87,14$	$T_{med_geral} = 51,04$ $T_{dp_geral} = 3,61$ $T_{min_geral} = 47,91$ $T_{max_geral} = 64,98$	$T_{med_geral} = 49,69$ $T_{dp_geral} = 1,72$ $T_{min_geral} = 48,24$ $T_{max_geral} = 57,76$
Vizinhança Totalística 3 atributos	$T_{med_geral} = 54,03$ $T_{dp_geral} = 3,43$ $T_{min_geral} = 49,60$ $T_{max_geral} = 64,90$	$T_{med_geral} = 53,48$ $T_{dp_geral} = 4,21$ $T_{min_geral} = 48,26$ $T_{max_geral} = 72,16$	$T_{med_geral} = 53,05$ $T_{dp_geral} = 2,55$ $T_{min_geral} = 50,06$ $T_{max_geral} = 61,54$
Vizinhança Totalística Nível dinâmico	$T_{med_geral} = 53,02$ $T_{dp_geral} = 2,82$ $T_{min_geral} = 50,14$ $T_{max_geral} = 62,80$	$T_{med_geral} = 51,49$ $T_{dp_geral} = 2,11$ $T_{min_geral} = 47,80$ $T_{max_geral} = 59,32$	$T_{med_geral} = 52,52$ $T_{dp_geral} = 1,18$ $T_{min_geral} = 51,24$ $T_{max_geral} = 56,84$

Tab. 6.15: Aplicação das regras descobertas com o modo de operação paralelo de AC, a Evolução Simples, Conjunta e AGC no escalonamento das 50 variações aleatórias do Gauss18.

Capítulo 7

Conclusões e Trabalhos Futuros

7.1 Conclusões

Ainda é um grande desafio o problema do escalonamento de tarefas em um sistema multiprocessador. A maioria dos algoritmos apresentados na literatura são seqüenciais e mostram-se pouco eficientes no escalonamento de diversos programas paralelos. Por outro lado, o algoritmo de escalonamento baseado em autômatos celulares (ACs) proposto em [94] é uma alternativa promissora na resolução desse tipo de problema. Em diversos trabalhos publicados, o algoritmo de escalonamento baseado em ACs foi apto a descobrir regras de transição capazes de resolver o problema para um grafo de programa dado. No entanto, muito pouco foi publicado sobre a reutilização das regras descobertas para um determinado grafo de programa, no escalonamento de outros grafos.

O objetivo principal dessa dissertação foi a investigação da capacidade de generalização das regras de ACs descobertas pelo algoritmo de escalonamento proposto em [94]. A execução desse algoritmo exige um considerável esforço computacional, principalmente na fase de aprendizagem de regras, onde é utilizado um AG na busca por regras de transição de ACs aptas a escalonar. Em decorrência desse esforço, entendemos que a utilização dos ACs para o escalonamento em um sistema multiprocessador só se justifica se as regras descobertas puderem ser de fato reutilizadas no escalonamento de diferentes programas paralelos.

O escalonador de tarefas em multiprocessadores discutido nessa dissertação é fortemente baseado no modelo proposto em [94]. A principal modificação investigada refere-se à fase de aprendizagem

de regras, na qual um AG é utilizado na busca de regras de ACs capazes de escalonar tarefas em uma arquitetura paralela.

Avaliamos três modelos de vizinhança de AC propostos para o algoritmo de escalonamento: a *Vizinhança Linear* proposta em [102], a *Vizinhança Seleccionada* e a *Vizinhança Totalística*, ambas propostas em [94]. Os principais programas paralelos (grafos de programa) utilizados nos experimentos iniciais foram o Gauss18, o *g18*, o *g40*, o *tree15* e o *intree15* [96, 102, 103, 104].

Nos experimentos realizados com o modelo de vizinhança linear, ao avaliarmos a capacidade de generalização das regras de ACs, aplicando-se as regras descobertas para um grafo de programa específico, no escalonamento dos demais grafos, percebemos que resultados razoáveis foram obtidos apenas entre os grafos de programa com uma estrutura regular (*g18*, *g40*, *tree15* e *intree15*). Quando as regras descobertas para um grafo com estrutura regular foram utilizadas para o Gauss18, e vice-versa, resultados satisfatórios não foram obtidos. Ao aplicarmos as regras descobertas para o Gauss18 no escalonamento de variações desse mesmo grafo, observamos que alguma capacidade de generalização essas regras possuem, porém, ainda há muito espaço para que a mesma seja aprimorada. Após os experimentos com a vizinhança linear, acreditamos que este modelo de vizinhança não é o mais adequado para a representação da estrutura do grafo de programa, uma vez que ele não captura as relações entre as tarefas expressas no grafo. Diante disso, nosso foco voltou-se exclusivamente para os modelos de vizinhança não-lineares (seleccionada e totalística).

Para avaliar a capacidade de generalização das regras descobertas com a utilização dos modelos de vizinhança não-lineares, aplicamos as regras descobertas para um grafo de programa específico no escalonamento de outros grafos de programa. Assim como nos experimentos realizados com o modelo de vizinhança linear, também observamos que resultados razoáveis foram obtidos somente entre os grafos de programa com estrutura regular. Assim, concluímos que o grafo de programa mais “difícil” de escalonar, dentre os que avaliamos, é o Gauss18, provavelmente devido à sua irregularidade. Devido à dificuldade de escalonamento apresentada pelo Gauss18, decidimos intensificar os estudos com este grafo de programa e suas variações. Os melhores resultados experimentais publicados para o Gauss18 foram obtidos com a utilização do modo de operação sequencial de AC [93, 94, 95, 96, 97, 98, 99]. Assim, embora não seja o modo de operação mais eficiente do ponto de vista prático, foi com ele que iniciamos nossos estudos. Ainda utilizando a mesma abordagem pro-

posta em [94], que chamamos de *Evolução Simples*, experimentos foram realizados com quatro tipos diferentes de vizinhança: vizinhança selecionada com três atributos (co-nível estático, custo computacional e custo de comunicação), a vizinhança selecionada com o atributo nível dinâmico, a vizinhança totalística com três atributos e a vizinhança totalística com o atributo nível dinâmico. Nesses experimentos concluímos que o modelo de vizinhança que melhor se adequou ao escalonamento do Gauss18 foi a vizinhança selecionada com três atributos, que corresponde ao modelo utilizado por Seredynski e colegas [96]. Entretanto, ao avaliarmos a capacidade de generalização, as regras evoluídas que apresentaram valores mais satisfatórios foram as que utilizaram o modelo de vizinhança totalística com o nível dinâmico. A utilização apenas do nível dinâmico como atributo de formação da vizinhança não-linear, seja ela selecionada ou totalística, não havia sido relatada anteriormente na literatura. Acreditamos que isso decorre do fato de que o nível dinâmico é um atributo que força o cálculo do mesmo a cada atualização de célula do AC, o que acarreta um grande aumento no tempo de processamento.

Objetivando o aperfeiçoamento da capacidade de generalização das regras evoluídas, decidimos nos concentrar em avaliar novas estratégias para a fase de aprendizagem das regras. Duas novas abordagens foram avaliadas nos modelos não-lineares de vizinhança: a *Evolução Conjunta* e a *Coevolução*.

Na nova abordagem *Evolução Conjunta* proposta nessa dissertação, a aprendizagem evolutiva deve ser baseada não somente na informação do grafo de programa que se deseja escalonar, mas também em pequenas variações do mesmo. Para avaliar a capacidade de generalização das regras, as mesmas foram utilizadas no escalonamento de quinze variações do Gauss18. O escalonamento realizado pelas regras obtidas através da *Evolução Conjunta* foi superior ao realizado pelas regras obtidas pela estratégia empregada em [94] (*Evolução Simples*). Apenas para o modelo de experimento “Vizinhança Totalística com o atributo nível dinâmico” a *Evolução Conjunta* não se mostrou eficaz. Entretanto, esse foi o modelo que retornou o melhor resultado de generalização com a *Evolução Simples*. Assim, concluímos que esse é o modelo, dentre os avaliados, que possui uma melhor capacidade de generalização na *Evolução Simples* por um lado, mas que não foi beneficiado pela *Evolução Conjunta*, por outro lado. Dentre os outros modelos, o que utilizou a vizinhança selecionada e três atributos foi o que mais se beneficiou da estratégia evolução conjunta, reduzindo o tempo

de execução das variações do Gauss18 em pelo menos 6 unidades de tempo, em média.

Na abordagem coevolutiva que propusemos para a fase de aprendizagem do algoritmo de escalonamento, a aprendizagem envolve o Gauss18 e outros nove grafos gerados aleatoriamente. Para avaliar a capacidade de generalização das regras, as mesmas foram utilizadas no escalonamento de cinquenta variações do Gauss18 geradas aleatoriamente. As regras descobertas com essa abordagem apresentaram melhores resultados quando comparadas às abordagens Evolução Simples e Conjunta, em três dos quatro modelos avaliados. Os melhores resultados foram obtidos com a “Vizinhança Seleccionada com três atributos diferentes”.

Numa análise geral dos experimentos envolvendo o modo de operação seqüencial, o experimento com a “Vizinhança Totalística com o atributo nível dinâmico” utilizando-se a abordagem Evolução Simples mostrou-se tão bom quanto o melhor experimento com a abordagem coevolutiva (Vizinhança Seleccionada com três atributos diferentes). No entanto, o tempo de processamento do algoritmo de escalonamento é consideravelmente menor no experimento com a abordagem coevolutiva, onde foram utilizados atributos estáticos, cujo valor é calculado apenas uma vez durante todo o processamento. Quando utilizado o atributo nível dinâmico, o tempo de processamento do algoritmo tende a ser maior, uma vez que o valor desse atributo pode ser calculado várias vezes durante o processamento.

Consideramos o modo de operação paralelo de AC fundamental para o algoritmo de escalonamento baseado em ACs, uma vez que o paralelismo é uma característica importante dos ACs, especialmente se pensarmos em termos de tempo de processamento de um escalonador baseado em ACs. Assim, realizamos experimentos com as abordagens Evolução Simples, Conjunta e Coevolutiva para os quatro modelos de experimento citados anteriormente. Com a Evolução Simples, o único modelo de experimento que foi apto a descobrir regras capazes de escalonar o Gauss18 foi o modelo “Vizinhança Seleccionada com três atributos”. Com a Evolução Conjunta, onde foram utilizadas quinze variações do Gauss18 para a avaliação das regras, a capacidade de generalização das regras foi melhorada em todos os modelos de experimento, sendo o melhor resultado obtido com o modelo “Vizinhança Seleccionada com o atributo nível dinâmico”. Um resultado também importante e que nos chamou atenção foi a aplicação das regras descobertas com a Evolução Simples e com a Evolução Conjunta para escalonar o próprio Gauss18 no modelo com a vizinhança selecionada e atributo nível dinâmico, no modo de operação paralelo. A melhor regra descoberta com a Evolução Conjunta foi

melhor que a melhor regra descoberta com a Evolução Simples. Nós entendemos que a “dificuldade” de adaptabilidade das regras imposta pela Evolução Conjunta no algoritmo de escalonamento baseado em AC, contribuiu para a descoberta de regras com um maior poder de escalonamento para o próprio Gauss18, mesmo sendo o Gauss18 utilizado na Evolução Simples durante todo o processo evolutivo.

Ao avaliarmos a capacidade de generalização das regras descobertas com as abordagens Simples, Conjunta e Coevolutiva em cinquenta variações aleatórias do Gauss18, no modo de operação paralelo, observamos que a coevolução foi melhor que a Evolução Simples em todos os modelos de experimento, e melhor que a Evolução Conjunta em três dos quatro modelos, sendo um pouco inferior apenas no modelo “Vizinhança Totalística com atributo nível dinâmico”.

Numa análise mais geral, tanto no modo paralelo quanto no seqüencial, as regras de ACs obtidas com a Evolução Simples que apresentaram uma melhor capacidade de generalização foram as descobertas com a vizinhança totalística e o atributo nível dinâmico. Para os experimentos com as abordagens conjunta e coevolutiva, os melhores resultados foram obtidos com a vizinhança selecionada. Na literatura, os melhores resultados para o Gauss18 foram obtidos com a utilização da vizinhança selecionada.

Com relação ao tempo de processamento do algoritmo de escalonamento baseado em ACs, concluímos que os experimentos que utilizam três atributos estáticos para a formação da vizinhança apresentam um tempo significativamente menor, quando comparados aos experimentos que utilizam o atributo nível dinâmico.

Comparando-se os modos de operação de AC, percebemos que os melhores resultados foram obtidos com o modo de operação seqüencial, porém, o modo paralelo demanda um menor esforço computacional. Aparentemente, o modo de operação paralelo foi menos explorado na literatura, exatamente pelo fato de não retornar resultados tão bons quanto o modo seqüencial. Entretanto, com a utilização das abordagens evolução conjunta e coevolutiva, não só foi possível melhorar a capacidade de generalização das regras, como viabilizou-se a obtenção de regras melhores para o próprio grafo evoluído.

Como conclusão final, entendemos que a utilização das abordagens conjunta e coevolutiva, principais contribuições desse trabalho, permitiram a melhoria da capacidade de generalização das regras descobertas pelo algoritmo de escalonamento baseado em ACs. No entanto, acreditamos que ainda

existe espaço para a elaboração de novas abordagens que possam contribuir ainda mais para a melhoria dessa capacidade, fundamental para o algoritmo proposto em [94].

7.2 Trabalhos Futuros

Os experimentos com a nossa abordagem coevolutiva e apresentados na Seção 6.2, foram conduzidos de forma a ajustar os parâmetros para o modelo de vizinhança selecionada com três atributos (co-nível estático, custo computacional e custo de comunicação). A escolha por este modelo ocorreu por dois motivos: 1) ele apresentou bons resultados para o Gauss18 nas abordagens simples e conjunta; 2) a utilização dos três atributos estáticos reduz o tempo de processamento do algoritmo. Assim, propomos como próximos experimentos, o ajuste dos parâmetros do algoritmo para os demais modelos de vizinhança (“Vizinhança Selecionada com atributo nível dinâmico”, “Vizinhança Totalística com três atributos” e “Vizinhança Totalística com atributo nível dinâmico”). Estes experimentos nos ajudarão a encontrar regras de ACs ainda melhores, ou concluir que o melhor modelo de vizinhança para o Gauss18 é de fato a selecionada com três atributos.

Na literatura e nos experimentos que reproduzimos com os modelos de vizinhança não-lineares propostos em [94], percebemos que, para grafos de programa com uma estrutura irregular, como o Gauss18, melhores resultados foram obtidos com a vizinhança selecionada e, para grafos com uma estrutura regular (*g18*, *g40*, *tree15*, *intree15* etc.), a vizinhança totalística mostrou-se mais adequada. Uma nova linha de investigação possível é a criação de um novo modelo de vizinhança de AC que englobe características de ambas as vizinhanças e que obtenha bons resultados para diferentes grafos de programa, independente da regularidade da estrutura dos mesmos.

Apresentamos também como proposta a elaboração de novas abordagens para a fase de aprendizagem do algoritmo de escalonamento baseado em ACs. Novas abordagens que vislumbramos no momento: (i) a utilização dos parâmetros de previsão do comportamento dos ACs (λ , Sensitividade, Propagação da Atividade e outros) como métrica auxiliar para a busca evolutiva; (ii) a alteração da função de aptidão proposta em [94], utilizando-se a estratégia de Recurso Compartilhado (*Resource Sharing*); (iii) outras abordagens evolutivas: os AGCs cooperativos, os AGs multi-objetivos e a Programação Genética.

Após alguns experimentos iniciais com a vizinhança linear, entendemos que esse modelo não expressa as relações de proximidade entre as tarefas do grafo de programa e nossa atenção voltou-se para os modelos de vizinhança não-linear, os quais expressam melhor tais relações. Realizados os experimentos com os modelos de vizinhança não-lineares e as abordagens simples, conjunta e coevolutiva, realizamos um experimento com a abordagem Evolução Conjunta e a vizinhança linear, descrito no Apêndice C. Os resultados obtidos foram promissores, porém inferiores aos obtidos com os modelos de vizinhança não-lineares. Dentre os modelos de vizinhança apresentados nessa dissertação, o que exige menor esforço computacional é o linear. Assim, propomos explorá-lo e criar um novo modelo linear que procure refletir melhor a relação entre as tarefas do grafo de programa.

Referências Bibliográficas

- [1] E. Aarts & J. Korst. *Simulated Annealing and Boltzmann Machines*. John Wiley & Sons, Chichester, U.K., 1989.
- [2] T. F. Abdelzaher & K. G. Shin. Optimal combined task and message scheduling in distributed real-time systems. In *IEEE Real-Time Systems Symposium*, pg. 162–171, 1995.
- [3] I. Ahmad & Y. K. Kwok. A parallel approach for multiprocessor scheduling. *ipps*, 00:289, 1995.
- [4] D. Andre, III Bennett, & J. Koza. Discovery by genetic programming of a cellular automata rule that is better than any known rule for the majority classification problem. In Stanford University, editor, *Genetic Programming 96*, 1996.
- [5] O. K. N. Asakura. Algoritmos genéticos coevolutivos na busca parametrizada de autômatos celulares unidimensionais. Dissertação de mestrado, Universidade Presbiteriana Mackenzie, São Paulo, SP, Brasil, 2001.
- [6] T. Bäck, U. Hammel, & H.-P. Schwefel. Evolutionary computation: comments on the history and current state. *IEEE Trans. Evolutionary Computation*, 1(1):3–17, 1997.
- [7] J. C. H. Barcellos. Algoritmos genéticos adaptativos: Um estudo comparativo. Dissertação de mestrado, Escola Politécnica da Universidade de São Paulo, São Paulo, SP, Brasil, 2000.
- [8] T. Bäck. Selective pressure in evolutionary algorithms: A characterization of selection mechanisms. Proc. 1st IEEE Conf. on Evolutionary Computation, IEEE Press, 1994.

- [9] E. R. Berlekamp, J. H. Conway, & R. K. Guy. *Winning Ways for Your Mathematical Plays*. Academic Press, New York, 1983.
- [10] H. Bersini & F. J. Varela. The immune recruitment mechanism: A selective evolutionary strategy. In *Proc. of the Fourth International Conference on Genetic Algorithms*, pg. 520–526, San Diego, CA, 1991.
- [11] P. Binder. A phase diagram for elementary cellular automata. *Complex Systems*, 7:241–247, 1993.
- [12] P. Binder. Parametric ordering of complex systems. *Physical Review E*, 1994.
- [13] J. Blazewicz. *Scheduling in Computer and Manufacturing Systems*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1996.
- [14] T. Blickle. *Theory of Evolutionary Algorithms and Application to System Synthesis*. PhD thesis, Swiss Federal Institute of Technology, Zurich, November 1996.
- [15] T. Blickle & L. Thiele. A comparison of selection schemes used in genetic algorithms. Technical Report 11, Gloriastrasse 35, 8092 Zurich, Switzerland, 1995.
- [16] P. Brucker. *Scheduling Algorithms*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
- [17] P. Callahan. What is the game of life? Disponível em <http://www.math.com/students/wonders/life/life.html>, Último acesso em 10 de Outubro de 2006.
- [18] R. E. Castro. Otimização de estruturas com multi-objetivos via algoritmos genéticos. Dissertação de mestrado, Universidade Federal do Rio de Janeiro, Rio de Janeiro, RJ, Brasil, 2001.
- [19] Ph. Chretienne. Task scheduling over distributed memory machines. In M. Cosnard, P. Quinton, M. Raynal, & Y. Robert, editors, *Proc. of Inter. Workshop on Parallel and Distributed Algorithms.*, pg. 165–176, North Holland, 1989.
- [20] E. F. Codd. *Cellular Automata*. Academic Press, Inc., Orlando, FL, USA, 1968.

- [21] A. P. A. da Silva. Tutorial on genetic algorithms. *Learning And Nonlinear Models, SBRN, Brasil*, 1:43–58, 2002.
- [22] R. Das, J. P. Crutchfield, M. Mitchell, & J. E. Hanson. Evolving globally synchronized cellular automata. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pg. 336–343, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- [23] R. Das, M. Mitchell, & J. P. Crutchfield. A genetic algorithm discovers particle-based computation in cellular automata. In *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pg. 344–353, London, UK, 1994. Springer-Verlag.
- [24] Y. Davidor. *Genetic Algorithms and Robotics*. World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1991.
- [25] L. Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, New York, New York, 1991.
- [26] L. N. de Castro & J. Timmis. *Artificial Immune Systems: A New Computational Intelligence Approach*. Springer-Verlag, London, 2002.
- [27] G. M. B. de Oliveira. Dinâmica e evolução de autômatos celulares unidimensionais. Tese de Doutorado. Instituto Tecnológico de Aeronáutica (ITA), São José dos Campos, 1999.
- [28] G. M. B. de Oliveira. Autômatos celulares: aspectos dinâmicos e computacionais. In Ricardo de Oliveira Anido & Paulo César Masiero, editors, *III Jornada de Mini-cursos em Inteligência Artificial (MCIA)*. : Sociedade Brasileira de Computação, volume 8, pg. 297–345, 2003.
- [29] G. M. B. de Oliveira, A. Coelho, & L. Monteiro. Criptografia baseada em autômatos celulares com sensibilidade bidirecional. *ENIA2003: Encontro Brasileiro de Inteligência Artificial*, 2003.
- [30] G. M. B. de Oliveira, A. R. Coelho, & L. H. A. Monteiro. Cellular automata cryptographic model based on bi-directional toggle rules. *International Journal of Modern Physics C*, 15(8):1061–1068, 2004.

- [31] G. M. B. de Oliveira, P. P. B. de Oliveira, & N. Omar. Evolving solutions of density classification task in 1d cellular automata, guided by parameters that estimate their dynamic behavior. In *Proceedings of the Artificial Life 2000 VII*, 2000.
- [32] G. M. B. de Oliveira, P. P. B. de Oliveira, & N. Omar. Guidelines for dynamics-based parameterization of one-dimensional cellular automata rule spaces. *Complexity*, 6(2):63–71, 2000.
- [33] G. M. B. de Oliveira, P. P. B. de Oliveira, & N. Omar. Definition and applications of a five-parameter characterization of one-dimensional cellular automata rule space. *Artificial Life*, 7(3):277–301, 2001.
- [34] G. M. B. de Oliveira, P. P. B. de Oliveira, & N. Omar. Improving genetic search for one-dimensional cellular automata, using heuristics related to their dynamic behavior forecast. In *Proceedings of the 2001 Congress on Evolutionary Computation CEC2001*, pg. 348–355, COEX, World Trade Center, 159 Samseong-dong, Gangnam-gu, Seoul, Korea, 27-30 May 2001. IEEE Press.
- [35] H. El-Rewini & T. G. Lewis. Scheduling parallel program tasks onto arbitrary target machines. *J. Parallel Distrib. Comput.*, 9(2):138–153, 1990.
- [36] H. El-Rewini, T. G. Lewis, & H. H. Ali. *Task scheduling in parallel and distributed systems*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.
- [37] A. C. R. S. Filho. Sisne - laboratório de sistemas neurais. Disponível em <http://neuron.ffc.lrp.usp.br/2004-I-IPEII-aula9.pdf>, Último acesso em 23 de Outubro de 2006.
- [38] M. Fuentes & M. Kuperman. Cellular automata and epidemiological models with spatial dependence. *Physica A*, 267:471–486, 1999.
- [39] M. R. Garey & D. S. Johnson. *Computers and Intractability. A Guide to the Theory of NP-Completeness*. A Series of Books in the Mathematical Sciences. Freeman And Company, 1979. GAR m2 79:1 1.Ex.

- [40] A. Gerasoulis & T. Yang. A comparison of clustering heuristics for scheduling directed acycle graphs on multiprocessors. *Journal of Parallel Distributed Computation*, 16(4):276–291, 1992.
- [41] A. Gerasoulis & T. Yang. Scheduling program task graphs on MIMD architectures. In R. Paige, J. Reif, & R. Wachter, editors, *Parallel Algorithm Derivation and Program Transformation*, pg. 153–186. Kluwer Academic Publishers, 1993.
- [42] F. Glover & M. Laguna. *Tabu Search*. Kluwer Academic Publishers, Norwell, MA, USA, 1997.
- [43] D. E. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
- [44] D. E. Goldberg & K. Deb. A comparative analysis of selection schemes used in genetic algorithms. In G. Rawlins, editor, *Foundations of Genetic Algorithms 1*, pg. 69–93. Morgan Kaufmann Publishers, 1991.
- [45] A. D. Gongora. O que é inteligência artificial? Disponível em <http://www.pr.gov.br/batebyte/edicoes/2002/bb119/estagiario.htm>, Último acesso em 28 de Novembro de 2006.
- [46] G. W. Greenwood, A. Gupta, & K. McSweeney. Scheduling tasks in multiprocessor systems using evolutionary strategies. In *International Conference on Evolutionary Computation*, pg. 345–349, 1994.
- [47] H. Gutowitz. Cryptography with dynamical systems. In N. Boccara, E. Goles, S. Martinez, & P. Picco, editors, *Cellular Automata and Cooperative Phenomena*, pg. 237–274. Kluwer Academic Publishers, 1993.
- [48] D. Hillis. Co-evolving parasites improves simulated evolution as an optimization procedure. *Physica D*, 42(1-3):228–234, 1991.
- [49] J. H. Holland. *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA, 1975.

- [50] J. H. Holland & J. H. Miller. Artificial adaptive agents in economic theory. *American Economic Review*, 81(2):365–71, May 1991. available at <http://ideas.repec.org/a/aea/aecrev/v81y1991i2p365-71.html>.
- [51] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. pg. 457–464, 1988.
- [52] C.-J. Hou & K. G. Shin. Replication and allocation of task modules in distributed real-time systems. In *Proc. of the IEEE Int. Symposium on Fault-Tolerant Computing*, pg. 26–35, 1994.
- [53] E. S. H. Hou, N. Ansari, & H. Ren. A genetic algorithm for multiprocessor scheduling. *IEEE Trans. Parallel Distrib. Syst.*, 5(2):113–120, 1994.
- [54] A. Jalilvand & S. Khanmohammadi. Task scheduling in manufacturing systems based on an efficient branch and bound algorithm. In *Proceedings of the 2004 IEEE Conference on Robotics, Automation and Mechatronics, Singapore, 1-3 December, 2004*, pg. 271–276, December 2004. InternalNote: Submitted by: jalilvanda@yahoo.com.
- [55] J. Jonsson & K. G. Shin. A parametrized branch-and-bound strategy for scheduling precedence-constrained tasks on a multiprocessor system. In *ICPP '97: Proceedings of the international Conference on Parallel Processing*, pg. 158–165, Washington, DC, USA, 1997. IEEE Computer Society.
- [56] H. Juillé & J. B. Pollack. Coevolving the “ideal” trainer: Application to the discovery of cellular automata rules. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, & R. L. Riolo, editors, *Genetic Programming Conference (GP-98)*, San Mateo, CA, July 1998. Morgan Kaufmann.
- [57] S. Kim & J. Browne. A general approach to mapping of parallel computations upon multiprocessor architectures. In *Proc. of Int. Conf. on Parallel Processing*, volume 2, pg. 1–8, University Park, PA, USA, Aug 1988.
- [58] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

- [59] N. Krasnogor, W. E. Hart, J. Smith, & D. A. Pelta. Protein structure prediction with evolutionary algorithms. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, & Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conference*, volume 2, pg. 1596–1601, Orlando, Florida, USA, 13-17 1999. Morgan Kaufmann.
- [60] Y. K. Kwok & I. Ahmad. Benchmarking the task graph scheduling algorithms. In *Proceedings 1998 IPPS/SPDP Symposium*, pg. 531–537, Washington, DC, USA, 1998. IEEE Computer Society.
- [61] M. Land & R. K. Belew. No perfect two-state cellular automata for density classification exists. *Physica Review Letters*, 74(25):5148–5150, Jun 1995.
- [62] C. Langton. Artificial life. In *Lecture in Complex Systems. SFI Studies in the Sciences of Complexity*, volume 4. Addison-Wesley, 1992.
- [63] C. G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1-3):12–37, 1990.
- [64] S. Levy. *Artificial life: the quest for a new creation*. Random House Inc., New York, NY, USA, 1992.
- [65] W. Li. Phenomenology of non-local cellular automata. *Journal of Statistical Physics*, 68(5-6), 1992.
- [66] W. Li & N. Packard. The structure of the elementary cellular automata rule space. *Complex Systems*, 4(3):281–297, 1990.
- [67] M. Markus & I. Kusch. Cellular automata for modelling the shell pigmentation of molluscs. *Journal of Biological Systems*, 3:999–1011, 1995.
- [68] M. Mitchell. Computation in cellular automata. to appear in *Non-standard computation*, Gramß et. al. (eds.), 1996.
- [69] M. Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge, MA, USA, 1996.

- [70] M. Mitchell, J. Crutchfield, & R. Das. Evolving cellular automata with genetic algorithms: A review of recent work. In *Proceedings of the First International Conference on Evolutionary Computation and its Applications (EvCA'96)*, volume 5, Moscow, 1996.
- [71] M. Mitchell, J. P. Crutchfield, & P. T. Hraber. Evolving cellular automata to perform computations: mechanisms and impediments. In *Proceedings of the Oji international seminar on Complex systems : from complex dynamical systems to sciences of artificial reality*, pg. 361–391, New York, NY, USA, 1994. Elsevier North-Holland, Inc.
- [72] M. Mitchell, P. T. Hraber, & J. P. Crutchfield. Revisiting the edge of chaos: Evolving cellular automata to perform computations. Technical Report Santa Fe Institute Working Paper 93-03-014, 1993.
- [73] T. Mitchell. *Machine Learning*. McGraw-Hill, N.Y., 1997.
- [74] J. V. Neumann. *Theory of Self-Reproducing Automata*. University of Illinois Press, Champaign, IL, USA, 1966.
- [75] M. Nourani, C. Papachristou, & Y. Takefuji. A neural network based algorithm for the scheduling problem in high-level synthesis. In *EURO-DAC '92: Proceedings of the conference on European design automation*, pg. 341–346, Los Alamitos, CA, USA, 1992. IEEE Computer Society Press.
- [76] S. K. Pal & P. P. Wang, editors. *Genetic Algorithms for Pattern Recognition*. CRC Press, Inc., Boca Raton, FL, USA, 1996.
- [77] J. Paredis. Co-evolutionary constraint satisfaction. In *PPSN III: Proceedings of the International Conference on Evolutionary Computation. The Third Conference on Parallel Problem Solving from Nature*, pg. 46–55, London, UK, 1994. Springer-Verlag.
- [78] J. Paredis. Steps towards co-evolutionary classification networks. In R. A. Brooks & P. Maes, editors, *Artificial Life IV, Proceedings of the fourth International Workshop on the Synthesis and Simulation of Living Systems.*, pg. 359–365. MIT Press, 1994.

- [79] J. Paredis. The symbiotic evolution of solutions and their representations. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pg. 359–365, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.
- [80] J. Paredis. Coevolutionary computation. *Artificial Life Journal*, 2(3), 1996.
- [81] J. Paredis. Coevolving cellular automata: Be aware of the red queen. In Thomas Bäck, editor, *Proceedings of the Seventh International Conference on Genetic Algorithms (ICGA97)*, San Francisco, CA, 1997. Morgan Kaufmann.
- [82] D.-T. Peng & K. G. Shin. Static allocation of periodic tasks with precedence constraints in distributed real-time systems. In M. Cosnard, P. Quinton, M. Raynal, & Y. Robert, editors, *Proc. of the IEEE Int. Conf. on Distributed Computing Systems*, pg. 190–198, New Port Beach, California, 1989.
- [83] M. A. Potter & K. De Jong. A cooperative coevolutionary approach to function optimization. In Yuval Davidor, Hans-Paul Schwefel, & Reinhard Männer, editors, *Parallel Problem Solving from Nature – PPSN III*, pg. 249–257, Berlin, 1994. Springer.
- [84] M. A. Potter & K. A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evolutionary Computation*, 8(1):1–29, 2000.
- [85] N. J. Radcliffe & P. D. Surry. Fitness variance of formae and performance prediction. In L. Darrell Whitley & Michael D. Vose, editors, *Foundations of Genetic Algorithms 3*, pg. 51–72, San Mateo, CA, 1995. Morgan Kaufmann Publishers.
- [86] S. J. Russell & P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.
- [87] S. Salleh & A. Y. Zomaya. Multiprocessor scheduling using mean-field annealing. In *IPPS/SPDP Workshops*, pg. 288–296, 1998.
- [88] O. Salvador. Introdução a Algoritmos Genéticos. Disponível em <http://descartes.ucpel.tche.br/WFC/2002/03-ag.pdf>, Último acesso em 17 de Dezembro de 2006.

- [89] D. D. Santos, D. Pasotto, & G. M. B. de Oliveira. Estudo comparativo de algoritmos genéticos aplicados ao escalonamento de tarefas. In *Caderno de Pós-Graduação em Eng. Elétrica*, volume 4, pg. 99–112. Universidade Presbiteriana Mackenzie, 2004.
- [90] V. Sarkar. *Partitioning and Scheduling Parallel Programs for Multiprocessors*. MIT Press, Cambridge, MA, USA, 1989.
- [91] A. Schoneveld. *Parallel complex systems simulation*, 1999.
- [92] F. Seredynski. Competitive coevolutionary multi-agent systems: The application to mapping and scheduling problems. *J. Parallel Distrib. Comput.*, 47(1):39–57, 1997.
- [93] F. Seredynski. Discovery with genetic algorithm scheduling strategies for cellular automata. In *Parallel Problem Solving from Nature - PPSN*, pg. 643–652, 1998.
- [94] F. Seredynski. Scheduling tasks of a parallel program in two-processor systems with use of cellular automata. *Future Generation Computer Systems*, 14(5-6):351–364, 1998.
- [95] F. Seredynski. Scheduling tasks of a parallel program in two-processor systems with use of cellular automata. In *IPPS/SPDP Workshops*, pg. 261–269, 1998.
- [96] F. Seredynski. Evolving cellular automata-based algorithms for multiprocessor scheduling. In S. Olariu A. Zomaya, F. Ercal, editor, *Solutions to Parallel and Distributed Computing Problems: Lessons from Biological Sciences*, Wiley Series on Parallel and Distributed Computing, pg. 179–207, New York, 2001. Wiley.
- [97] F. Seredynski & A. Swiecicka. Immune-like system approach to cellular automata-based scheduling. In *PPAM*, pg. 626–633, 2001.
- [98] F. Seredynski, A. Swiecicka, & A. Y. Zomaya. Discovery of parallel scheduling algorithms in cellular automata-based systems. In *IPDPS*, pg. 132, 2001.
- [99] F. Seredynski & A. Y. Zomaya. Sequential and parallel cellular automata-based scheduling algorithms. *IEEE Trans. Parallel Distrib. Syst.*, 13(10):1009–1023, 2002.

- [100] M. Sipper. Co-evolving non-uniform cellular automata to perform computations. *Physica D*, 92(3-4):193–208, 1996.
- [101] M. Sipper. *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
- [102] A. Swiecicka & F. Seredynski. Cellular automata approach to scheduling problem. In *PA-RELEC '00: Proceedings of the International Conference on Parallel Computing in Electrical Engineering*, pg. 29, Washington, DC, USA, 2000. IEEE Computer Society.
- [103] A. Swiecicka & F. Seredynski. Applying cellular automata in multiprocessor scheduling. In *PARELEC '02: Proceedings of the International Conference on Parallel Computing in Electrical Engineering*, pg. 177, Washington, DC, USA, 2002. IEEE Computer Society.
- [104] A. Swiecicka & F. Seredynski. Multiprocessor scheduling and rescheduling with use of cellular automata and artificial immune system support. *IEEE Trans. Parallel Distrib. Syst.*, 17(3):253–262, 2006. Fellow-Albert Y. Zomaya.
- [105] G. Sywerda. Uniform crossover in genetic algorithms. In *Proceedings of the third international conference on Genetic algorithms*, pg. 2–9, San Francisco, CA, USA, 1989. Morgan Kaufmann Publishers Inc.
- [106] M. C. S. Takiguti. Utilização de algoritmos genéticos multi-objetivos na mineração de regras precisas e interessantes. Dissertação de mestrado, Universidade Presbiteriana Mackenzie, São Paulo, SP, Brasil, 2003.
- [107] J. Tanomaru. Motivação, fundamentos e aplicações de algoritmos genéticos. In *II Congresso Brasileiro de Redes Neurais*, Curitiba, 1995.
- [108] C. E. Taylor & D. Jefferson. Artificial life as a tool for biological inquiry. *Artificial Life*, 1(1-2):1–14, 1994.
- [109] A. M. Turing. Computing machinery and intelligence. *INTELLIGENCE.MIND*, 59:433, 1950.

- [110] P. A. Vargas, C. L. Filho, & F. J. V. Zuben. Redução de perdas em redes de distribuição de energia elétrica através de sistemas classificadores. In *Revista Controle & Automação*, volume 14, Julho 2003.
- [111] G. Vichniac. Simulating physics with cellular automata. *Physica D*, 10:96–116, 1984.
- [112] P. Vidica, L. B. da Silveira, & G. M. B. de Oliveira. Escalonamento de tarefas baseado na evolução conjunta de autômatos celulares com vizinhança totalística. In *Congresso Brasileiro de Automática (CBA)*. CBA, 2006.
- [113] P. Vidica & G. M. B. de Oliveira. Cellular automata-based scheduling: A new approach to improve generalization ability of evolved rules. In *Brazilian Symposium on Artificial Neural Networks (SBRN)*. IEEE Press, 2006.
- [114] J. Werfel, M. Mitchell, & J. P. Crutchfield. Resource sharing and coevolution in evolving cellular automata. *IEEE-EC*, 4(4), November 2000.
- [115] J. Whale, L. Neubert, J. Esser, & M. Schreckenberg. A cellular automaton traffic flow model for online simulation of traffic. *Parallel Computing*, 27(5):719–735, 2001.
- [116] Wikipedia. Gaussian elimination. Disponível em http://en.wikipedia.org/wiki/Gaussian_elimination, Último acesso em 12 de Outubro de 2006.
- [117] M. Wojtowicz. Cellular automata gallery. Disponível em http://psoup.math.wisc.edu/mcell/ca_gallery.html, Último acesso em 10 de Outubro de 2006.
- [118] S. Wolfram. Cellular automata. *Los Alamos Science*, 9:2–21, 1983.
- [119] S. Wolfram. Statistical mechanics of cellular automata. *Rev. Modern Physics*, 55:601–644, 1983.
- [120] S. Wolfram. Computation theory of cellular automata. *Communications in Mathematical Physics*, 96:15–57, 1984.
- [121] S. Wolfram. Universality and complexity in cellular automata. *Physica D*, 10:1–35, 1984.

- [122] S. Wolfram. Cryptography with cellular automata. *Proceedings of Crypto '85*, pg. 429–432, 1985.
- [123] S. Wolfram. Complex systems theory. In D. Pines, editor, *Emerging Syntheses in Science*. Addison-Wesley, Redwood City, California, 1988.
- [124] S. Wolfram. *Cellular automata and complexity*. Addison-Wesley, U.S.A, 1994.
- [125] S. Wolfram. *A New Kind of Science*. Wolfram Media, U.S.A, May 2002.
- [126] M. Wu & D. Gajski. A programming aid for hypercube architectures. *Journal of Supercomputing*, 2:349–372, 1988.
- [127] T. Yang & A. Gerasoulis. A fast static scheduling algorithm for dags on an unbounded number of processors. In *Supercomputing '91: Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pg. 633–642, New York, NY, USA, 1991. ACM Press.
- [128] I. Yepes. Projeto ISIS: Uma incursão aos algoritmos genéticos. <http://www.geocities.com/igoryepes/index.htm>, Último acesso em 10 de Outubro de 2006.

Apêndice A

Intervalo de Confiança e Testes de Hipóteses

O *Intervalo de Confiança* e os *Testes de Hipóteses* [73] são métodos estatísticos utilizados na interpretação de dados experimentais obtidos.

A.1 Intervalo de Confiança

Sendo Θ um **parâmetro populacional verdadeiro** (mas desconhecido), o intervalo de confiança fornece um intervalo de valores plausíveis para este parâmetro baseado nos dados amostrais.

A.1.1 Intervalos de confiança de 95% para uma média

Um intervalo de confiança de 95% para um parâmetro populacional fornece um intervalo no qual estaríamos 95% confiantes de cobertura do verdadeiro valor do parâmetro.

Seja n o tamanho amostral, μ a média populacional, σ o desvio padrão e $\frac{\sigma}{\sqrt{n}}$ o **erro padrão** (SE) da média. O cálculo do intervalo de confiança de 95% para um média é feito da seguinte forma:

$$(\bar{x} - 1.96 \times \frac{\sigma}{\sqrt{n}}, \bar{x} + 1.96 \times \frac{\sigma}{\sqrt{n}}) \quad (\text{A.1})$$

Isto significa que intervalos da forma acima citada conterão a verdadeira média populacional μ 95% das vezes.

Um problema com a construção de tais intervalos é que não se sabe o verdadeiro desvio padrão populacional. Para grandes tamanhos amostrais, contudo, o desvio padrão amostral s será uma boa estimativa de σ . Portanto, pode-se substituir σ por s de modo que pode-se calcular o erro padrão (SE) como

$$SE = \frac{s}{\sqrt{n}} \quad (\text{A.2})$$

e um intervalo de confiança de aproximadamente 95% para μ é:

$$\left(\bar{x} - 1.96 \times \frac{s}{\sqrt{n}}, \bar{x} + 1.96 \times \frac{s}{\sqrt{n}} \right) \quad (\text{A.3})$$

Para amostras pequenas ($n \leq 30$), onde s é uma estimativa menos confiável de μ , o intervalo de confiança deve ser calculado de forma diferente. Ao invés de usar o valor 1.96, deve ser utilizado um valor maior para refletir a redução da confiança. Este valor é obtido através da **tabela de distribuição t** (Tabela A.1), tomando-se o valor correspondente à linha $r = n - 1$ graus de liberdade. Assim, um intervalo de confiança exato é

$$\left(\bar{x} - t_{(n-1.005)} \times \frac{s}{\sqrt{n}}, \bar{x} + t_{(n-1.005)} \times \frac{s}{\sqrt{n}} \right) \quad (\text{A.4})$$

É importante observar que quanto menor o valor de n , maior o valor de t e, a medida que o valor de n aumenta, o valor de t torna-se próximo a 1.96.

A.2 Testes de Hipóteses

Os testes de hipóteses têm como objetivo decidir, com base na informação fornecida pelos dados de uma amostra, sobre a aceitação ou não de uma dada hipótese.

Em geral, intervalos de confiança são a forma mais informativa de apresentar os achados princi-

	80%	90%	95%	99%	99.90%
1	3.078	6.314	12.706	63.657	636.619
2	1.886	2.920	4.303	9.925	31.599
3	1.638	2.353	3.182	5.841	12.924
4	1.533	2.132	2.776	4.604	8.610
5	1.476	2.015	2.571	4.032	6.869
6	1.440	1.943	2.447	3.707	5.959
7	1.415	1.895	2.365	3.499	5.408
8	1.397	1.860	2.306	3.355	5.041
9	1.383	1.833	2.262	3.250	4.781
10	1.372	1.812	2.228	3.169	4.587
11	1.363	1.796	2.201	3.106	4.437
12	1.356	1.782	2.179	3.055	4.318
13	1.350	1.771	2.160	3.012	4.221
14	1.345	1.761	2.145	2.977	4.140
15	1.341	1.753	2.131	2.947	4.073
16	1.337	1.746	2.120	2.921	4.015
17	1.333	1.740	2.110	2.898	3.965
18	1.330	1.734	2.101	2.878	3.922
19	1.328	1.729	2.093	2.861	3.883
20	1.325	1.725	2.086	2.845	3.850
21	1.323	1.721	2.080	2.831	3.819
22	1.321	1.717	2.074	2.819	3.792
23	1.319	1.714	2.069	2.807	3.768
24	1.318	1.711	2.064	2.797	3.745
25	1.316	1.708	2.060	2.787	3.725
26	1.315	1.706	2.056	2.779	3.707
27	1.314	1.703	2.052	2.771	3.690
28	1.313	1.701	2.048	2.763	3.674
29	1.311	1.699	2.045	2.756	3.659
30	1.310	1.697	2.042	2.750	3.646
40	1.303	1.684	2.021	2.704	3.551
50	1.299	1.676	2.009	2.678	3.496
60	1.296	1.671	2.000	2.660	3.460
70	1.294	1.667	1.994	2.648	3.435
80	1.292	1.664	1.990	2.639	3.416
90	1.291	1.662	1.987	2.632	3.402
100	1.290	1.660	1.984	2.626	3.390
∞	1.282	1.645	1.960	2.576	3.291

Tab. A.1: Tabela de distribuição de t

país de um estudo. Contudo, algumas vezes existe um particular interesse em decidir sobre a verdade ou não de uma hipótese específica (se dois grupos têm a mesma média ou não, ou se o parâmetro populacional tem um valor em particular ou não), e o teste de hipóteses fornece-nos a estrutura para que façamos isto.

A.2.1 Procedimento geral de testes

Para realizar o teste de hipóteses, o procedimento [73] geral é:

1. Estabeleça a hipótese nula H_0 e a hipótese alternativa H_1 .
2. Decida qual o teste a ser usado, checando se este é válido para o seu problema.
3. Calcule a estatística de teste, T .
4. Encontre a probabilidade (p -valor) de observar um valor tão extremo ou maior do que T se a hipótese nula é de fato verdadeira. Você precisará se referir aos valores críticos nas tabelas estatísticas as quais fornecem p -valores correspondendo aos valores das estatística de teste.
5. Avalie a força da evidência contra H_0 . (Quanto menor p -valor, tanto mais evidência contra a hipótese nula.) Se necessário, decida se esta é evidência suficiente para rejeitar (ou não rejeitar) a hipótese nula.
6. Estabeleça as conclusões e interpretação dos resultados.

O p -valor é a probabilidade de observar dados tão extremos quanto os obtidos se a hipótese nula é verdadeira.

A.2.2 Teste para uma média

Os passos para se calcular o teste de hipóteses para uma média são [73]:

1. Estabeleça a hipótese nula, $H_0: \mu = \mu_0$, e a hipótese alternativa $H_1: \mu \neq \mu_0$.
2. Calcule a média amostral $\hat{\mu} = \bar{x}$ e o desvio padrão amostral s .

3. Calcule o erro padrão, $SE = s/\sqrt{n}$.
4. Calcule a estatística de teste $t = (\hat{\mu} - \mu_0)/SE$. Este é o número de erros padrão que $\hat{\mu}$ dista do valor de hipótese μ_0 .
5. Encontre o p -valor da distribuição t , com $r = n - 1$ graus de liberdade, da tabela usando os valores absolutos da estatística de teste.
6. Estabeleça conclusões e interprete os resultados.

Para um melhor entendimento deste método, é descrito a seguir um exemplo [37] que ilustra o mesmo.

A.2.3 Um exemplo de testes de hipóteses

Para que um produtor de suco de laranja possa vender o seu produto nos supermercados ele tem que garantir que o peso médio da embalagem de 1 litro de suco seja igual a 1080 g. Para fazer isso, o produtor recolhe uma amostra de 20 embalagens de 1 litro de suco da sua fábrica e mede o peso de cada uma delas, obtendo os dados apresentados na Tabela A.2.

Embalagem	Peso	Embalagem	Peso
1	1050	11	1065
2	1008	12	1073
3	1067	13	1030
4	1078	14	1046
5	1065	15	1060
6	1015	16	1059
7	1070	17	1069
8	1064	18	1067
9	1066	19	1047
10	1072	20	1072

Tab. A.2: Exemplo do cálculo do teste de hipótese: dados amostrais de 20 embalagens de 1 litro de suco

O produtor precisa decidir, com base nos dados da sua amostra, se a embalagem de 1 litro de suco de laranja que ele produz tem um peso médio diferente de 1080 g ou não.

Para fazer isso, o produtor formula o que se chama de **hipótese nula**, denominada por **H0**. A sua hipótese nula é a seguinte:

H0 As embalagens de 1 litro de suco de laranja que eu produzo têm peso médio igual a 1080 g.

O problema do produtor é agora saber se, com base nos seus dados amostrais, pode-se rejeitar ou aceitar a hipótese nula.

Repare que a hipótese nula é equivalente a supor que o produto está de acordo com as normas exigidas pelos supermercados. A hipótese contrária à hipótese nula é chamada de **hipótese alternativa** e é representada por **H1**.

Portanto:

$$H0: \mu = 1080 \text{ g}$$

$$H1: \mu \neq 1080 \text{ g}$$

O produtor poderia pensar que para testar a sua hipótese nula bastaria usar o valor médio para a sua amostra ($\bar{x}$) e verificar se ele é igual a 1080g ou não. Entretanto, por causa da *variação amostral*, mesmo que o valor médio do peso das suas embalagens de 1 litro de suco de laranja seja igual a 1080g, uma amostra particular escolhida ao acaso pode ter valor médio do peso maior ou menor que 1080g (Figura A.1). O que deve ser decidido é se o valor de $\bar{x}$ obtido para a amostra é *significativamente* diferente de 1080 g.

Além da hipótese nula, o produtor também vai assumir que:

- A distribuição dos pesos das embalagens de 1 litro de suco de laranja é aproximadamente normal. Isto implica que a distribuição amostral dos pesos médios de amostras de 20 embalagens é normal.
- O desvio padrão da distribuição dos pesos das embalagens de 1 litro de suco de laranja é conhecido e vale 25 g. Isto implica que o desvio padrão da distribuição amostral dos pesos médios de amostras de 20 embalagens é igual a $\sigma_{\bar{x}} = \frac{\sigma}{\sqrt{n}} = \frac{25}{\sqrt{20}} = 5,6 \text{ g}$.

Portanto, com a hipótese nula e as duas considerações feitas acima, a distribuição amostral dos pesos médios de amostras de 20 embalagens está apresentada na Figura A.2, onde também está marcado

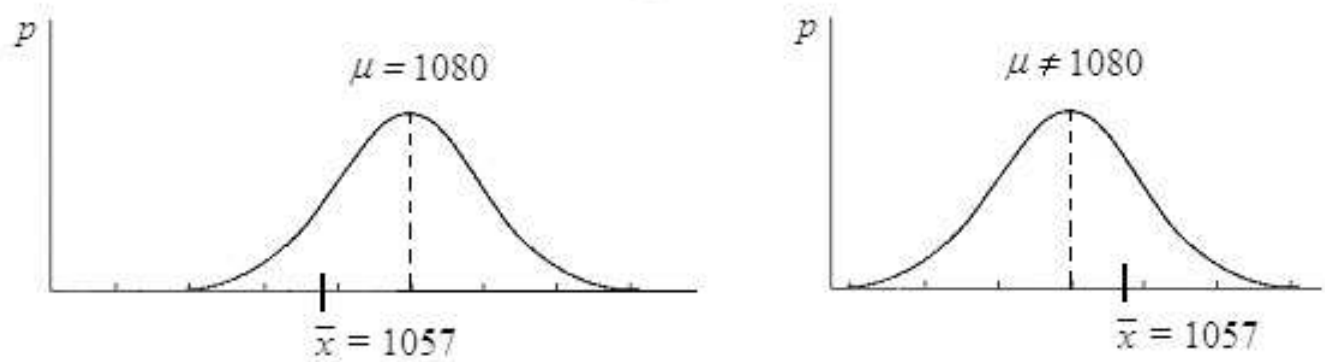


Fig. A.1: Duas distribuições amostrais de pesos de embalagens de 1 litro de suco de laranja. As duas são obtidas com amostras de 20 embalagens retiradas das suas respectivas populações. Uma tem média igual a 1080g e a outra não tem. Note que tanto para um caso como para o outro podem haver amostras com peso médio $\bar{x} = 1057$ g.

o valor médio para a amostra selecionada, $\bar{x} = 1057$ g. A pergunta que o produtor pode fazer agora é: qual a probabilidade de se obter uma amostra com valor médio $\bar{x} = 1057$ g para uma distribuição normal como a do gráfico, com média $\mu_{\bar{x}} = 1080$ g?

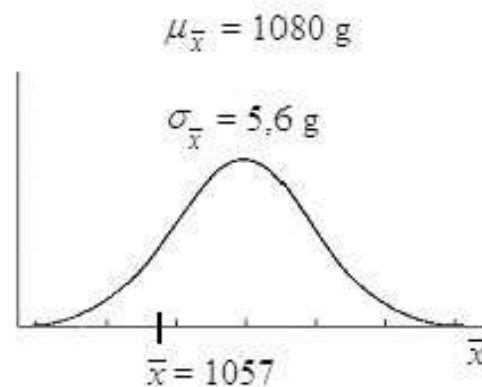


Fig. A.2: Distribuição amostral dos pesos médios de amostras de 20 embalagens.

Segundo o método dos testes de hipóteses, a pergunta que se faz é um pouco diferente: “Qual a probabilidade de se obter um valor de $\bar{x}$ tão ou mais distante da média $\mu_{\bar{x}}$ quanto $\bar{x} = 1057$ g?”.

Graficamente, a resposta a esta pergunta é a soma das áreas pintadas do gráfico da Figura A.3, chamada de **valor p**.

Para se calcular o valor p, basta transformar a distribuição normal do exemplo para a distribuição normal padrão, obter o correspondente valor de z, calcular as áreas acima de -z e abaixo de z e

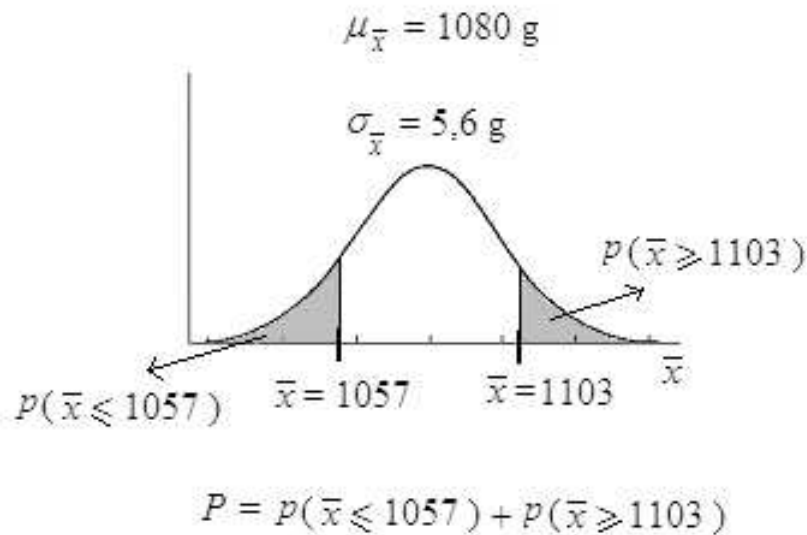


Fig. A.3: Distribuição amostral com o valor de $\bar{x} = 1057$ g para a sua amostra aleatória de 20 embalagens.

somá-las.

$$z = \frac{1057 - 1080}{5,6} = -4,1.$$

Consultando uma tabela, vê-se que a área abaixo de z vale 0,00002. Portanto, a área acima de $-z$ também vale 0,00002 e o valor p é:

$$P = 2 \times 0,00002 = 0,00004.$$

Portanto, se o peso médio das embalagens de 1 litro de suco de laranja for igual a 1080g, a probabilidade de que se obtenha, ao acaso, uma amostra de 20 embalagens com peso médio igual ou mais distante da média quanto 1057g é de 0,00004 (0,004%).

Este é um valor muito baixo, o que indica que, ou se obteve uma amostra muito rara, ou a hipótese nula é falsa e o peso médio das embalagens não é de 1080g.

Face a estas duas possibilidades e ao valor extremamente baixo de p , o produtor decide pela falsidade da hipótese nula. Ou seja, ele conclui que o peso médio das embalagens de 1 litro de suco de laranja produzidas por ele é diferente de 1080g.

É claro que é possível que ele esteja errado e que o peso médio das embalagens seja, de fato, igual a 1080g. Porém, devido ao baixíssimo valor p obtido a conclusão do produtor parece ter sido a mais razoável.

Suponhamos agora que, ao invés de 1057g o produtor tivesse obtido o valor $\bar{x} = 1078$ g para a sua amostra aleatória de 20 embalagens. Neste caso, teríamos um caso como o descrito pelo gráfico abaixo:

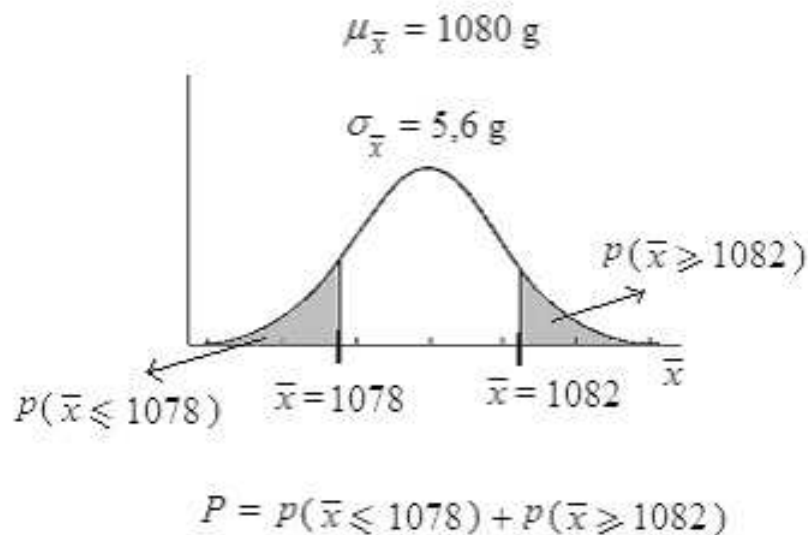


Fig. A.4: Distribuição amostral com o valor de $\bar{x} = 1078$ g para a sua amostra aleatória de 20 embalagens.

Realizando os mesmos cálculos que os feitos para o caso anterior, obtemos agora um valor $p = 2 \times 0,1406 = 0,2812$, ou 28,12%.

Neste segundo caso, como $p = 0,28$ não é um valor desprezível o produtor decide não rejeitar a hipótese nula. Note que ele não está aceitando a hipótese nula; apenas não possui argumentos para provar que ela é falsa. Neste caso, a não rejeição da hipótese nula é a conclusão mais razoável a ser tomada.

Nos dois exemplos dados, foi bem fácil recusar a hipótese nula quando $p = 0,00004$ e foi bem fácil não recusá-la quando $p = 0,28$. Esta decisão não seria tão fácil se o valor p fosse alguma coisa intermediária, por exemplo $p = 0,1$. É importante, então, definir um valor padrão de p para servir de corte entre rejeição ou não rejeição da hipótese nula.

O valor de p mais comumente aceito para servir de divisor entre rejeição e não rejeição de H_0 é $p = 0,05$ (5%).

Este valor de 0,05 é designado pela letra grega α e é chamado de *nível de significância*. Pode-se escolher um outro valor para α , como $\alpha = 0,1$ ou 0,01, mas o mais aceito é $\alpha = 0,05$.

Para $\alpha = 0,05$, o produtor tem 5% de chances de errar quando rejeita a hipótese nula, pois 5% das amostras de 20 embalagens retiradas aleatoriamente da população de embalagens produzidas terão peso médio igual ou mais distante de 1080 g de modo a fazer $p = 0,05$.

Apêndice B

Principais resultados publicados por Seredynski e colegas

Nas próximas seções descreveremos os principais resultados publicados por Seredynski e seus colegas, com a utilização do escalonador descrito na Seção 5.1.4. Chamamos esses resultados de “experimentos originais”. Em todos os experimentos descritos, a política de escalonamento foi “a tarefa com maior nível dinâmico é executada primeiro”. Os experimentos reproduzidos, baseados nos originais, estão descritos no Capítulo 5.

B.1 Resultados com o modelo de vizinhança linear

Resultados recentes [102, 103] apresentaram um algoritmo para o escalonamento de tarefas baseado em ACs com o modelo de vizinhança linear descrito na Seção 5.1.2.

Em [102] foram realizados experimentos utilizando-se o grafo de programa Gauss18, onde foram avaliados três modos de operação no AC (paralelo, seqüencial e aleatório) e três valores de raio r diferentes (1, 2 e 3), resultando em regras de transição de ACs de tamanho 8, 32 e 128 bits, respectivamente. Os parâmetros utilizados para o AC nos experimentos foram $CI_s = 50$ e $M = 50$. Para o raio 1, os parâmetros do AG foram $P = 50$, $E = 25$, $p_m = 0,375$ e $G = 50$. Para o raio 2 foram utilizados $P = 100$, $E = 50$, $p_m = 0,093$ e $G = 100$. Finalmente, para o raio 3 foram utilizados $P = 200$, $E = 100$, $p_m = 0,023$ e $G = 1.000$. Para os três valores de raio foi utilizado o parâmetro $p_c = 0,50$. Os

melhores resultados foram obtidos com $r = 3$ e com o modo de operação seqüencial do AC, onde o AG descobriu regras capazes de levar o AC, a partir de todas configurações iniciais testadas, para uma configuração que representa o tempo total de execução ótimo $T = 44,0$. Para os demais raios e modos de operação, a melhor regra de transição de AC descoberta na última geração leva o AC a convergir, após M passos de tempo, para uma configuração que apresenta o tempo total de execução no intervalo $50 \leq T \leq 60$.

Para testar a qualidade das melhores regras obtidas de raio 3 e modo de operação seqüencial, 1.000 novas CIs aleatórias foram utilizadas na fase de operação. Neste teste, verificou-se que cerca de 100 regras (50% da população) foram capazes de encontrar um escalonamento ótimo a partir de cada alocação inicial aleatória. Ou seja, cada uma destas 100 regras foi aplicada, por 50 passos de tempo (M), em cada uma das 1.000 alocações iniciais das tarefas do Gauss18 nos processadores P0 e P1, convergindo todas elas para alocações finais que representam $T = 44,0$. Assim, os autores concluíram que os resultados dos experimentos conduzidos foram muito promissores. Porém, perguntas como: “Quão gerais são as regras descobertas?”, “Como melhorar a eficiência do algoritmo?”, “As regras descobertas podem ser aplicadas para o resolver o problema do escalonamento para outros grafos de programas diferentes?”, foram feitas e não foram respondidas, sendo consideradas, pelos autores, assunto para trabalhos futuros.

Em [103] foram realizados experimentos utilizando-se apenas o modo de operação paralelo (síncrono) e os grafos de programa *g18* (Figura 5.7a) e *intree15* (Figura 5.7d). Nesse AG, o tipo de *crossover* utilizado foi o de dois-pontos.

No primeiro experimento, o grafo utilizado foi o *g18*. Foram utilizados dois valores de raio: 1 e 2. Para cada AC na população, os parâmetros utilizados foram $CIs = 50$ e $M = 50$. Para o AG utilizado, os parâmetros configurados foram $P = 50$, $E = 25$, $G = 100$, $p_c = 0,50$ e $p_m = 0,03$. O valor do raio que conseguiu convergir o AC para o valor ótimo de T foi $r = 2$. O AG não precisou mais que 20 gerações para descobrir uma regra de transição de AC capaz de convergir cada uma das 50 CIs para alocações das tarefas que correspondem ao valor de T igual a 46,0. Em uma análise mais apurada, constatou-se que o AC necessitou de cerca de 25 passos de tempo (M) para convergir (evoluir) para uma alocação de tarefas que representa um escalonamento ótimo para o *g18*. A Figura B.1a apresenta um escalonamento ótimo das tarefas do *g18*. A qualidade das regras descobertas, para o raio $r = 2$,

foi testada na fase de operação, onde 1.000 novos testes de problema foram gerados aleatoriamente e utilizados para avaliar as regras descobertas na última geração do AG. Nesta avaliação, foi constatado que, das 50 regras da população, 30 foram aptas a encontrar um escalonamento ótimo para todos os testes.

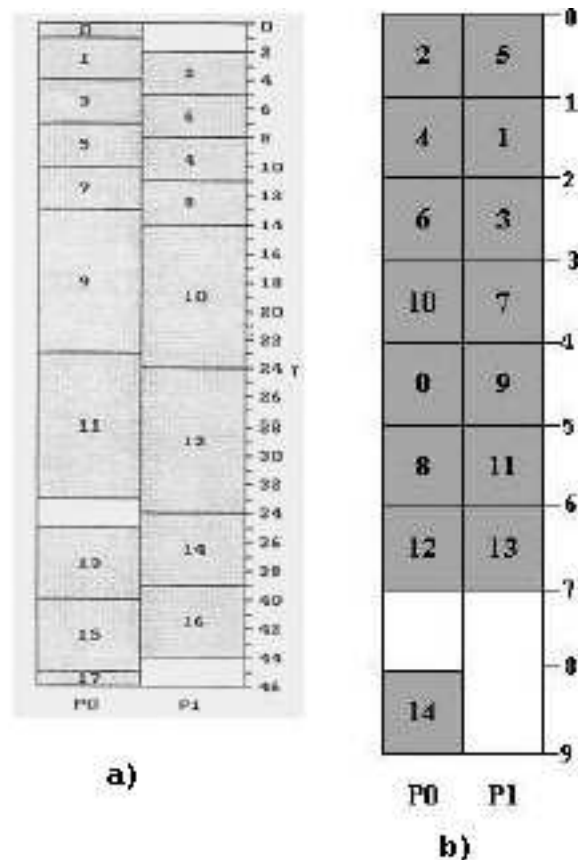


Fig. B.1: (a) Exemplo de escalonamento ótimo para o *g18*. (b) Exemplo de escalonamento ótimo para o *intree15* [103].

Em outro experimento, as regras descobertas para o grafo de programa *g18* foram utilizadas na fase de operação para encontrar soluções para o grafo de programa *intree15*. Assim, foram geradas alocações iniciais para o *intree15* nos processadores, e estas foram utilizadas para testar cada regra descoberta para o *g18*. A melhor média de T obtida foi igual a 9,0, correspondente ao T ótimo para o *intree15*. Isto significa que as melhores regras evoluídas para o *g18* encontraram, com sucesso, o escalonamento ótimo para cada uma das configurações iniciais do *intree15*. A Figura B.1b apresenta um escalonamento ótimo das tarefas do *intree15* nos processadores P0 e P1.

Swiecicka e Seredynski [103] concluíram que os resultados dos experimentos realizados também foram promissores. O AG foi apto a descobrir regras de transição de ACs capazes de resolver o problema do escalonamento para uma dada instância do problema. As regras descobertas para um dado grafo de programa foram usadas para encontrar o escalonamento ótimo em outro grafo de programa, o que os levou a propor, como trabalho futuro, um estudo sistemático do problema de reusabilidade das regras descobertas.

Em [97, 104], assim como nos experimentos relatados anteriormente, o propósito foi descobrir, com a utilização de um AG, regras de transição de ACs aptas a convergir, após alguns passos de tempo, qualquer alocação inicial (configuração inicial) de tarefas para uma alocação de tarefas que represente um escalonamento ótimo das tarefas em um sistema com dois processadores. Porém, o principal objetivo destes trabalhos foi estudar a possibilidade de reutilização das regras descobertas, isto é, utilizar as regras descobertas para um determinado grafo de programa em outros grafos de programa diferentes. Para isso, a proposta de Seredynski e Swiecicka foi a utilização de uma abordagem chamada de *sistemas imunológicos artificiais* (SIA) [26]. Nessa abordagem, as regras de transição evoluídas para um determinado grafo de programa são reutilizadas como população inicial do AG para um novo grafo de programa, que é semelhante ao grafo para o qual elas foram descobertas anteriormente. Assim, o objetivo da abordagem SIA é reaproveitar um conhecimento adquirido anteriormente pelas regras.

B.2 Resultados com os modelos de vizinhança não-linear e o modo de operação seqüencial

Resultados recentes [94, 95, 96, 99] apresentaram a utilização de modelos de vizinhança não-lineares pelo algoritmo de escalonamento baseado em AC.

Em [95] o modelo de vizinhança não-linear utilizado foi a *Vizinhança Seleccionada*, onde somente duas tarefas representativas de cada conjunto de tarefas *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)* são utilizadas na elaboração da vizinhança de uma célula associada a uma tarefa *k*. Os atributos utilizados para o conjunto de *predecessoras(k)*, *irmãs(k)* e *sucessoras(k)* foram o co-nível estático, o custo computacional e o custo de comunicação, respectivamente. A política de escalonamento

utilizada foi “a tarefa com maior nível dinâmico é executada primeiro”.

O grafo de programa utilizado no experimento foi o Gauss18. Utilizou-se no AG uma estratégia elitista de seleção para reprodução e reinserção e os parâmetros utilizados foram $P = 100$, $G = 100$, $p_c = 0,95$ e $p_m = 0,001$. O tipo de *crossover* utilizado foi o de ponto-simples e o tipo de mutação foi o de complemento do alelo (bit). A cada geração, quatro novas configurações iniciais do AC ($CIs = 4$) foram utilizadas na avaliação das regras. Para a evolução do AC foram utilizados 80 passos de tempo ($M = 80$). A elite E é formada por 5 indivíduos, e os 95 indivíduos restantes, na próxima geração, são formados utilizando-se os operadores *crossover* e mutação nos indivíduos da elite E . Para testar a qualidade das regras descobertas, as mesmas foram aplicadas em 100 novas configurações iniciais do Gauss18. Cerca de 29 regras da população final foram capazes de encontrar o valor ótimo de T para o Gauss18, em todas as 100 CIs avaliadas.

Em [94], alguns dos experimentos descritos foram realizados com o modelo de vizinhança não-linear *Vizinhança Seleccionada* e outros com a *Vizinhança Totalística*. Os principais grafos de programa utilizados foram o Gauss18, *g18* e *tree15*. Os parâmetros do AG e AC foram idênticos aos utilizados nos experimentos em [95], relatados anteriormente.

O primeiro experimento conduzido foi idêntico ao experimento realizado para o Gauss18 em [95], o qual descrevemos anteriormente. No segundo experimento, o grafo de programa utilizado foi o *g18*. O modelo de vizinhança do AC foi a *Vizinhança Totalística*, e a política de escalonamento adotada foi “a tarefa com o maior nível estático é executada primeiro” (Seção 4.1). O AG precisou de, aproximadamente, 20 gerações para descobrir uma regra capaz de levar a um escalonamento ótimo para o *g18* ($T = 46,0$). As regras descobertas para o *g18* foram avaliadas em 100 novas configurações iniciais na fase de operação. Cerca de 60% das regras descobertas foram capazes de levar o AC a convergir para uma configuração que leva a $T = 46,0$ [94]. Comparando-se a velocidade de convergência do AG e a eficiência das regras descobertas para o Gauss18 e o *g18*, os autores do trabalho chegaram a conclusão de que o grafo de programa *g18* é um problema mais fácil de resolver em relação ao Gauss18.

No terceiro e último experimento realizado em [94], o grafo de programa utilizado foi o *tree15*, sendo adotada a política de escalonamento “a tarefa com menor número de ordem é executada primeiro” (Seção 4.1). A vizinhança do AC utilizada foi a *Vizinhança Totalística*. Foram necessárias

apenas 5 gerações para o AG descobrir uma regra capaz de convergir o reticulado do AC para uma configuração que leva ao T ótimo igual a 9,0. Na fase de operação, cerca de 80% das regras descobertas na fase anterior foram aptas a encontrar soluções ótimas para o *tree15*. Estas regras descobertas também foram utilizadas na fase de operação para outros grafos de programa da família de árvores, os grafos *tree63*, que consiste de 63 tarefas, e o *tree127*, que consiste de 127 tarefas. Nos dois casos, as regras descobertas para o *tree15* foram capazes de encontrar escalonamentos ótimos.

Em [96] foi realizado um experimento utilizando-se o grafo de programa g40 e a *Vizinhança Seleccionada*. Nele, a estratégia de seleção elitista foi utilizada e os parâmetros do AG foram $P = 200$, $G = 200$, $p_c = 0,95$ e $p_m = 0,001$. Para o AC foram utilizados $M = 80$ e $CI_s = 5$. A política de escalonamento adotada foi “a tarefa com maior nível dinâmico é executada primeiro”. Regras de AC provendo um escalonamento ótimo ($T = 80,0$) foram encontradas após 160 gerações. As regras descobertas foram avaliadas na fase de operação em 100 novas CIs e cerca de 20 regras encontradas na fase de aprendizagem foram aptas a encontrar soluções ótimas na fase de operação para todas as configurações iniciais. Em outros experimentos [96] foi promovida a competição entre as regras descobertas para os grafos de programa na fase de aprendizagem de regras. Assim, regras descobertas para um determinado grafo de programa foram aplicadas para outros grafos de programa diferentes. As 25 melhores regras descobertas para o Gauss18 foram aplicadas, na fase de operação, para o *tree15*. Das 25 regras, cerca de 12 foram capazes de resolver o problema do escalonamento para o *tree15*. Da mesma forma, as 25 melhores regras descobertas para o *tree15* também foram aplicadas para o Gauss18. Nenhuma destas regras foi capaz de encontrar o escalonamento ótimo para o Gauss18, sendo o menor valor de T encontrado, aproximadamente, igual a 61,0. Os autores do trabalho, analisando os experimentos conduzidos com outras regras, concluíram que algumas regras encontradas para uma instância do problema são mais gerais que outras. Estas regras podem ser ordenadas de acordo com sua importância e podem ser usadas para resolver novos problemas.

Apêndice C

A abordagem Evolução Conjunta e o modelo de vizinhança linear

C.1 Experimento

Após termos apresentado alguns experimentos com o modelo de vizinhança linear de AC na Seção 5.5, originais e simulados (reproduzidos), voltamos a nossa atenção única e exclusivamente para os modelos de vizinhança não-lineares propostos em [94] e descritos na Seção 5.1.3. Dedicamos esta atenção a estes modelos de vizinhança porque entendemos que eles representam, de forma mais coerente, a estrutura de um grafo de programa. Por exemplo, seja o grafo de programa Gauss18. De acordo com o modelo de vizinhança linear, descrito na Seção 5.1.2, considerando, por exemplo, um valor de raio r igual a 3, a vizinhança da tarefa 6, que é associada à célula 6 do reticulado do AC, seria formada pelas tarefas 3, 4, 5, 6, 7, 8 e 9. Observando o grafo de programa (Figura 4.2), podemos ver que tarefas como 0, 1, 2, 10 e 11 que têm uma correlação com a tarefa 6 não são consideradas. Aumentando-se o valor do raio r para 6, as tarefas 0, 1, 2, 10, 11 e 12 passam a fazer parte da vizinhança da tarefa 6. Porém, além da tarefa 12 não ter uma relação direta com a tarefa 6, o tamanho da regra de transição de AC com $r = 6$ é 2^{13} , isto é, 8.192 bits, o que resulta em um espaço de regras de tamanho $2^{8.192}$. Além disso, para qualquer grafo de programa formado por 18 tarefas, independente de sua estrutura, a vizinhança da tarefa 6, por exemplo, será a mesma independente do grafo de programa, dependendo apenas do valor do raio assumido.

Nos modelos de vizinhança não-lineares, a estrutura não-linear do grafo de programa é representando por uma vizinhança não-linear de AC, onde para cada tarefa k do grafo de programa a vizinhança é formada pela própria tarefa k e os conjuntos de tarefas *predecessoras*(k), *irmãs*(k) e *sucessoras*(k). Assim, a vizinhança não depende do valor do raio (nestes modelos ele não é utilizado), considerando somente tarefas que têm relação com a tarefa em questão e o tamanho das regras de transição de AC nos modelos é fixo, sendo 250 bits para a *Vizinhança Seleccionada* e 128 bits para a *Vizinhança Totalística*.

Assim sendo, nas Seções 5.6 e 6.3, os experimentos foram conduzidos utilizando-se apenas os modelos de vizinhança não-lineares. Porém, como em alguns trabalhos [97, 102, 103, 104] o modelo de vizinhança linear apresentou bons resultados, resolvemos realizar também um experimento com a abordagem Evolução Conjunta (Seção 6.1) que propusemos para a fase de aprendizagem do algoritmo de escalonamento baseado em AC, objetivando melhorar a capacidade de generalização das regras de AC. Este experimento é descrito a seguir.

O principal grafo de programa utilizado no experimento foi o Gauss18. Para avaliar a capacidade de generalização das regras descobertas, utilizamos as quinze variações do Gauss18 descritas na Seção 5.3.

Utilizando a abordagem Evolução Conjunta, na fase de aprendizagem do algoritmo de escalonamento as regras de transição de AC foram evoluídas calculando-se seu valor de aptidão com base nos grafos de programa Gauss18, Gauss18_1, Gauss18_2, Gauss18_3, Gauss18_4 e Gauss18_5. Como mencionamos nos experimentos descritos anteriormente com a evolução conjunta, estas cinco variações foram escolhidos por serem aquelas com o maior número de modificações com relação ao grafo de programa original.

Os parâmetros utilizados para o AG e AC foram: $P = 100$, $E = 10$, $p_c = 0,90$, $p_m = 0,023$, $G = 100$, $CIs = 25$ e $M = 50$. O método de *crossover* utilizado foi o de um-ponto e o método de mutação foi o de complemento do alelo (bit). O tamanho de raio r escolhido foi igual a 3 e o modo de operação (atualização) de AC utilizado foi o sequencial. Escolhemos este valor de raio e este modo de operação porque na literatura e nos experimentos que simulamos, descritos na Seção 5.5.1, resultados ótimos só foram encontrados para o Gauss18, com a vizinhança linear, através deles.

Na Seção 5.5.1 apresentamos um experimento onde as melhores regras descobertas para o Gauss18

(Evolução Simples) foram aplicadas para as quinze variações deste grafo, para avaliar a capacidade de generalização das mesmas. Estas regras foram descobertas utilizando-se os mesmos parâmetros do AG e AC, valor de raio, métodos de *crossover* e mutação e modo de operação de AC acima citados e utilizados no experimento desta seção. Os resultados obtidos estão listados na Tabela 5.5 e, para facilitar o entendimento do leitor, resolvemos repeti-los na Tabela C.1. Comparando-se com os valores de referência de T para as quinze variações do Gauss18 listados na Tabela 5.3, apenas para os grafos de programa Gauss18_7 e Gauss18_10 existiram regras aptas a encontrar o escalonamento ótimo. Para os grafos de programa Gauss18_2, Gauss18_3, Gauss18_4, Gauss18_5 e Gauss18_11 nenhuma regra conseguiu encontrar boas soluções e, para os demais grafos, apenas soluções sub-ótimas foram encontradas. Nós calculamos o intervalo de confiança para os resultados obtidos e, estamos 95% confiantes que o valor médio de T é 49,20 para as quinze variações do Gauss18 e 47,30 para as dez variações, de Gauss18_6 à Gauss18_15.

Realizamos, então, trinta execuções do algoritmo de escalonamento utilizando a abordagem Evolução Conjunta. Na melhor delas, o valor médio de T para o melhor indivíduo da população na última geração foi igual a 295,40. Uma regra capaz de escalonar os seis grafos de programa presentes na fase de aprendizagem deve apresentar o valor médio de T igual a 282,0, segundo os valores de referência da Tabela C.1 (T_{AG_Ref}). A Figura C.1 apresenta a fase de aprendizagem para a melhor das trinta execuções.

Para avaliar a capacidade de generalização das regras descobertas utilizando-se a abordagem Evolução Conjunta, aplicamos, na fase de operação, estas regras para o escalonamento das cinco variações que participaram da fase de aprendizagem e de outras dez variações que não participaram desta fase. Para testar cada regra descoberta utilizamos 100 novos testes de problema (alocações iniciais das tarefas em P0 e P1). Os resultados encontrados estão listados na Tabela C.1. Comparando-se com os resultados obtidos com o experimento com a Evolução Simples (Tabela 5.5), em oito dos quinze grafos de programa, incluindo quatro dos cinco que participaram da fase de aprendizagem, o experimento com a Evolução Conjunta apresentou melhores resultados, isto é, os valores médios de T para oito grafos de programa foram melhores na Evolução Conjunta. Além disso, a média dos valores médios de T (coluna T_{med}) para cada grafo de programa foi igual a 49,20 com a Evolução Simples e 48,14 com a Evolução Conjunta. A média dos melhores valores de T (coluna T_{min}) foi também

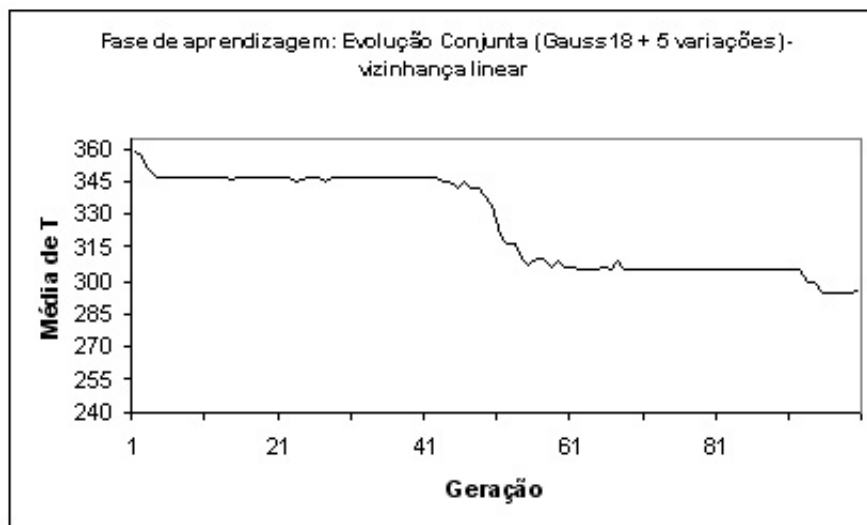


Fig. C.1: Fase de aprendizagem do algoritmo de escalonamento: evolução conjunta do Gauss18 e 5 variações deste grafo, considerando o modelo de vizinhança linear com o raio $r = 3$.

49,20 para a Evolução Simples e 48,13 para a Evolução Conjunta. Considerando apenas os grafos de programa que não participaram da fase de aprendizagem (Gauss18_6 à Gauss18_15), a Evolução Conjunta apresentou uma melhor de T em quatro deles, enquanto a Evolução Simples foi melhor em seis. A média, nas dez variações, foi 47,30 na Evolução Simples e 47,41 na Evolução Conjunta, sendo assim, resultados similares. Nós calculamos o intervalo de confiança também para os experimentos com a Evolução Conjunta. Nós estamos 95% confiantes de que a média do tempo total de execução T está no intervalo $[48,08; 48,20]$ considerando-se as quinze variações, e no intervalo $[47,35; 47,47]$ considerando-se as variações Gauss18_6 à Gauss18_15.

Nós também avaliamos o teste de hipótese nula para comparar os dois experimentos. Considerando as quinze variações, nós estamos 95% confiantes de que há uma melhoria, da Evolução Conjunta em relação a Evolução Simples, entre 0,99 e 1,11. Se considerarmos apenas as dez variações que não participaram da fase de aprendizagem, a melhoria não é verificada, sendo a Evolução Simples melhor entre 0,05 e 0,17. Neste caso, não podemos afirmar que a Evolução Conjunta descobriu regras mais gerais que as descobertas com a Evolução Simples. Logo, para o modelo de vizinhança linear, a abordagem Evolução Conjunta não apresentou resultados tão satisfatórios quanto os apresentados com os modelos de vizinhança não-lineares.

Grafo	T_{AG_Ref}	Evolução Simples				Evolução Conjunta			
	T_{med}	T_{med}	T_{dp}	T_{min}	T_{max}	T_{med}	T_{dp}	T_{min}	T_{max}
Gauss18_1	47,0	48,0	0,00	48,0	48,0	49,017	0,37	49,0	60,0
Gauss18_2	47,0	52,0	0,00	52,0	52,0	49,017	0,37	49,0	60,0
Gauss18_3	47,0	55,0	0,00	55,0	55,0	47,013	0,29	47,0	56,0
Gauss18_4	47,0	55,0	0,00	55,0	55,0	50,006	0,18	50,0	56,0
Gauss18_5	50,0	55,0	0,00	55,0	55,0	53,006	0,18	53,0	59,0
Gauss18_6	47,0	48,0	0,00	48,0	48,0	49,017	0,37	49,0	60,0
Gauss18_7	44,0	44,0	0,00	44,0	44,0	47,013	0,29	47,0	56,0
Gauss18_8	44,0	47,0	0,00	47,0	47,0	47,013	0,29	47,0	56,0
Gauss18_9	47,0	48,0	0,00	48,0	48,0	47,013	0,29	47,0	56,0
Gauss18_10	44,0	44,0	0,00	44,0	44,0	47,013	0,29	47,0	56,0
Gauss18_11	47,0	51,0	0,00	51,0	51,0	47,013	0,29	47,0	56,0
Gauss18_12	47,0	48,0	0,00	48,0	48,0	49,017	0,37	49,0	60,0
Gauss18_13	44,0	47,0	0,00	47,0	47,0	47,013	0,29	47,0	56,0
Gauss18_14	46,0	48,0	0,00	48,0	48,0	47,013	0,29	47,0	56,0
Gauss18_15	47,0	48,0	0,00	48,0	48,0	47,013	0,29	47,0	56,0
Média (10)	45,70	47,30	0,00	47,30	47,30	47,41	0,30	47,40	56,80
Média (15)	46,33	49,20	0,00	49,20	49,20	48,14	0,29	48,13	57,26

Tab. C.1: Fase de operação do algoritmo de escalonamento: Evolução Conjunta do Gauss18 mais 5 variações e modelo de vizinhança linear com raio $r = 3$.



Apêndice D

Alguns resultados detalhados

Este apêndice apresenta os tempos de execução obtidos ao aplicarmos as regras de ACs descobertas com as abordagens Evolução Simples, Conjunta e Coevolutiva em variações aleatórias do Gauss18. Os experimentos resumidos, relacionados a estes experimentos, foram apresentados no Capítulo 6.

Os experimentos foram realizados com os modelos de vizinhança selecionada e totalística, com o atributo nível dinâmico e também com três atributos diferentes (co-nível estático, custo computacional e custo de comunicação).

D.1 Modo de operação seqüencial de AC

D.1.1 Vizinhança Selecionada e três atributos

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	44,0	0,00	44,0	44,0	V18_20	53,37	2,11	53,0	66,0
V18_11	48,0	0,00	48,0	48,0	V18_21	60,31	7,50	53,0	83,0
V18_12	44,0	0,00	44,0	44,0	V18_22	54,94	1,00	54,0	56,0
V18_13	44,0	0,00	44,0	44,0	V18_23	49,10	0,70	49,0	54,0
V18_14	44,0	0,00	44,0	44,0	V18_24	47,0	0,00	47,0	47,0
V18_15	45,58	3,88	44,0	67,0	V18_25	54,38	2,80	53,0	74,0
V18_16	44,0	0,00	44,0	44,0	V18_26	51,36	4,04	48,0	65,0
V18_17	48,43	2,96	46,0	53,0	V18_27	44,0	0,00	44,0	44,0
V18_18	44,0	0,00	44,0	44,0	V18_28	44,0	0,00	44,0	44,0
V18_19	45,74	2,12	44,0	49,0	V18_29	59,0	0,00	59,0	59,0
Média	45,17	0,89	44,60	48,10	Média	51,74	1,81	50,40	59,20
V18_30	44,33	0,94	44,0	47,0	V18_40	46,05	0,21	46,0	47,0
V18_31	49,51	2,78	46,0	53,0	V18_41	62,05	6,83	53,0	72,0
V18_32	58,71	7,31	50,0	72,0	V18_42	51,08	2,00	49,0	53,0
V18_33	48,39	5,81	44,0	69,0	V18_43	56,88	5,78	50,0	70,0
V18_34	50,70	6,96	47,0	68,0	V18_44	47,25	5,01	44,0	61,0
V18_35	48,21	1,03	48,0	54,0	V18_45	56,10	7,97	48,0	67,0
V18_36	55,40	7,87	49,0	65,0	V18_46	49,72	5,43	48,0	73,0
V18_37	53,0	0,00	53,0	53,0	V18_47	53,10	0,70	53,0	58,0
V18_38	48,0	0,00	48,0	48,0	V18_48	57,97	6,04	52,0	70,0
V18_39	54,14	3,03	53,0	70,0	V18_49	62,75	6,88	53,0	82,0
Média	51,03	3,57	48,20	59,90	Média	54,29	4,68	49,60	65,30
V18_50	50,29	2,13	50,0	69,0	Média geral	51,38	2,98	48,76	60,56
V18_51	57,35	3,44	55,0	84,0					
V18_52	53,87	5,06	49,0	73,0					
V18_53	54,16	4,22	51,0	71,0					
V18_54	52,67	6,95	48,0	79,0					
V18_55	59,20	6,74	52,0	69,0					
V18_56	50,44	2,70	47,0	57,0					
V18_57	50,0	0,00	50,0	50,0					
V18_58	59,21	4,56	53,0	74,0					
V18_59	59,64	4,12	55,0	77,0					
Média	54,68	3,99	51,0	70,30					

Tab. D.1: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,0	0,00	49,0	49,0	V18_20	53,06	4,89	50,0	74,0
V18_11	53,0	0,00	53,0	53,0	V18_21	60,40	4,88	52,0	72,0
V18_12	49,0	0,00	49,0	49,0	V18_22	49,0	0,00	49,0	49,0
V18_13	49,0	0,00	49,0	49,0	V18_23	56,46	6,40	49,0	68,0
V18_14	48,98	0,20	47,0	49,0	V18_24	51,0	0,00	51,0	51,0
V18_15	49,0	0,00	49,0	49,0	V18_25	57,50	5,55	52,0	68,0
V18_16	48,98	0,20	49,0	49,0	V18_26	56,42	4,53	52,0	62,0
V18_17	49,0	0,00	49,0	49,0	V18_27	48,98	0,20	47,0	49,0
V18_18	49,0	0,00	49,0	49,0	V18_28	57,53	5,41	53,0	69,0
V18_19	54,95	3,53	49,0	59,0	V18_29	58,66	6,16	53,0	78,0
Média	49,99	0,39	49,20	50,40	Média	54,90	3,80	50,80	64,0
V18_30	57,87	6,22	49,0	71,0	V18_40	58,38	10,86	49,0	86,0
V18_31	49,0	0,00	49,0	49,0	V18_41	61,71	2,70	61,0	74,0
V18_32	56,37	6,65	50,0	75,0	V18_42	52,97	5,90	47,0	66,0
V18_33	48,60	6,17	46,0	80,0	V18_43	50,20	2,09	48,0	53,0
V18_34	53,75	4,51	49,0	70,0	V18_44	52,41	3,45	50,0	84,0
V18_35	46,0	0,00	46,0	46,0	V18_45	59,12	8,42	47,0	78,0
V18_36	54,0	0,00	54,0	54,0	V18_46	54,44	3,61	52,0	73,0
V18_37	49,25	1,43	49,0	58,0	V18_47	55,17	4,02	50,0	70,0
V18_38	49,62	4,00	47,0	66,0	V18_48	63,57	7,18	52,0	73,0
V18_39	56,26	7,40	49,0	72,0	V18_49	60,36	6,54	53,0	83,0
Média	52,07	3,63	48,80	64,10	Média	56,83	5,47	50,90	74,0
V18_50	55,57	3,29	55,0	76,0	Média geral	53,63	3,52	50,22	64,68
V18_51	57,06	3,13	53,0	61,0					
V18_52	47,18	3,39	46,0	72,0					
V18_53	60,55	5,34	56,0	81,0					
V18_54	51,79	6,60	49,0	77,0					
V18_55	52,66	3,57	49,0	58,0					
V18_56	49,0	0,00	49,0	49,0					
V18_57	56,56	6,00	54,0	83,0					
V18_58	49,88	3,62	49,0	69,0					
V18_59	63,60	8,59	54,0	83,0					
Média	54,38	4,35	51,40	70,90					

Tab. D.2: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Seleccionada com três atributos diferentes e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	45,59	1,50	44,0	47,0	V18_20	47,22	2,85	44,0	59,0
V18_11	48,58	1,00	48,0	56,0	V18_21	50,69	7,73	44,0	75,0
V18_12	44,16	1,12	44,0	52,0	V18_22	49,33	3,89	44,0	61,0
V18_13	45,20	1,66	44,0	52,0	V18_23	48,45	1,53	47,0	54,0
V18_14	45,07	1,57	44,0	52,0	V18_24	47,10	0,43	47,0	49,0
V18_15	44,21	1,22	44,0	52,0	V18_25	50,26	7,98	47,0	93,0
V18_16	44,0	0,00	44,0	44,0	V18_26	46,40	2,08	44,0	52,0
V18_17	44,35	1,65	44,0	55,0	V18_27	45,07	1,57	44,0	52,0
V18_18	47,87	9,99	44,0	94,0	V18_28	45,13	3,55	44,0	63,0
V18_19	46,59	2,21	44,0	55,0	V18_29	49,87	3,19	49,0	77,0
Média	45,56	2,19	44,40	55,90	Média	47,95	3,48	45,40	63,50
V18_30	45,81	2,90	44,0	58,0	V18_40	47,0	2,17	46,0	58,0
V18_31	44,35	1,65	44,0	55,0	V18_41	56,80	5,00	53,0	65,0
V18_32	48,80	3,12	47,0	61,0	V18_42	47,92	9,64	40,0	58,0
V18_33	44,77	3,84	44,0	79,0	V18_43	49,62	4,35	47,0	65,0
V18_34	47,0	0,00	47,0	47,0	V18_44	47,45	4,99	43,0	71,0
V18_35	47,16	1,61	45,0	63,0	V18_45	46,20	1,15	46,0	54,0
V18_36	49,39	1,86	49,0	66,0	V18_46	49,05	3,58	45,0	76,0
V18_37	53,17	3,20	47,0	59,0	V18_47	49,32	1,97	47,0	56,0
V18_38	48,84	1,45	44,0	57,0	V18_48	52,93	4,11	49,0	59,0
V18_39	50,81	3,91	44,0	59,0	V18_49	56,59	3,07	50,0	63,0
Média	48,01	2,35	45,50	60,40	Média	50,28	4,00	46,60	62,50
V18_50	53,64	4,72	50,0	66,0	Média geral	48,58	3,52	45,80	62,77
V18_51	51,72	7,92	46,0	69,0					
V18_52	45,78	4,42	43,0	83,0					
V18_53	52,48	3,34	51,0	63,0					
V18_54	51,22	5,25	47,0	68,0					
V18_55	52,59	9,82	45,0	89,0					
V18_56	50,83	3,86	47,0	62,0					
V18_57	49,97	2,85	49,0	77,0					
V18_58	52,05	11,48	44,0	83,0					
V18_59	51,17	2,15	49,0	56,0					
Média	51,14	5,58	47,10	71,60					

Tab. D.3: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Selecionada com três atributos diferentes e o modo de operação seqüencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
Gauss18_60	47,31	1,49	45,0	53,0
Gauss18_61	41,0	0,00	41,0	41,0
Gauss18_62	46,36	6,45	44,0	78,0
Gauss18_63	51,95	1,09	50,0	62,0
Gauss18_64	51,5	1,83	49,0	58,0
Gauss18_65	50,63	1,67	50,0	59,0
Gauss18_66	47,44	1,66	47,0	55,0
Gauss18_67	39,68	4,36	36,0	56,0
Gauss18_68	46,94	2,76	45,0	51,0
Gauss18_69	60,1	1,98	55,0	61,0
Média	48,29	2,32	46,20	57,40
Gauss18_70	47,46	1,07	46,0	50,0
Gauss18_71	54,62	5,09	51,0	78,0
Gauss18_72	67,69	10,69	54,0	82,0
Gauss18_73	45,6	2,52	44,0	54,0
Gauss18_74	49,13	0,98	49,0	58,0
Gauss18_75	54,16	0,55	54,0	56,0
Gauss18_76	45,96	3,23	44,0	58,0
Gauss18_77	45,93	3,91	39,0	59,0
Gauss18_78	43,19	4,22	40,0	56,0
Gauss18_79	56,88	4,75	51,0	91,0
Média	51,06	3,70	47,20	64,20
Gauss18_80	45,8	3,96	42,0	56,0
Gauss18_81	48,09	0,90	48,0	57,0
Gauss18_82	56,9	3,66	53,0	65,0
Gauss18_83	49,24	1,00	49,0	55,0
Gauss18_84	41,1	1,00	41,0	51,0
Gauss18_85	46,52	4,56	45,0	70,0
Gauss18_86	40,22	2,18	37,0	48,0
Gauss18_87	49,53	3,43	45,0	63,0
Gauss18_88	47,03	3,96	45,0	60,0
Gauss18_89	47,02	1,79	44,0	50,0
Média	47,14	2,64	44,90	57,50
Média Geral	48,83	2,88	46,10	59,70

Tab. D.4: Valores obtidos nas 30 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Selecionada com três atributos diferentes e o modo de operação seqüencial de AC.

D.1.2 Vizinhaça Selecionada e o atributo nível dinâmico

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	56,77	10,51	49,0	85,0	V18_20	43,01	0,09	43,0	44,0
V18_11	47,18	1,26	47,0	56,0	V18_21	70,09	6,22	60,0	83,0
V18_12	48,2	9,91	44,0	75,0	V18_22	44,18	1,02	44,0	50,0
V18_13	44,0	0,0	44,0	44,0	V18_23	55,66	7,25	47,0	66,0
V18_14	44,0	0,0	44,0	44,0	V18_24	45,0	0,0	45,0	45,0
V18_15	44,0	0,0	44,0	44,0	V18_25	71,37	14,76	55,0	94,0
V18_16	44,0	0,0	44,0	44,0	V18_26	46,44	8,85	44,0	85,0
V18_17	63,24	13,60	45,0	90,0	V18_27	44,0	0,0	44,0	44,0
V18_18	44,15	1,07	44,0	53,0	V18_28	52,14	14,04	44,0	82,0
V18_19	44,0	0,0	44,0	44,0	V18_29	64,93	8,07	51,0	90,0
Média	47,95	3,63	44,90	57,90	Média	53,68	6,03	47,70	68,30
V18_30	55,43	4,00	52,0	62,0	V18_40	55,45	3,98	52,0	62,0
V18_31	64,78	11,15	50,0	94,0	V18_41	60,52	6,14	53,0	85,0
V18_32	59,37	11,73	47,0	90,0	V18_42	53,68	3,49	50,0	57,0
V18_33	44,53	3,70	44,0	79,0	V18_43	62,74	4,44	53,0	76,0
V18_34	44,12	0,84	44,0	50,0	V18_44	68,78	12,00	46,0	92,0
V18_35	47,13	0,91	47,0	54,0	V18_45	50,07	1,23	48,0	53,0
V18_36	57,22	11,81	49,0	80,0	V18_46	57,08	1,84	47,0	62,0
V18_37	45,02	2,49	44,0	53,0	V18_47	60,87	14,98	47,0	86,0
V18_38	56,03	5,46	46,0	73,0	V18_48	59,13	14,42	48,0	80,0
V18_39	64,1	9,06	49,0	95,0	V18_49	63,05	8,46	53,0	94,0
Média	53,77	6,11	47,20	73,0	Média	59,13	7,09	49,70	74,70
V18_50	70,11	5,19	56,0	87,0	Média geral	54,97	6,05	47,50	70,70
V18_51	53,76	4,66	49,0	85,0					
V18_52	53,64	3,01	51,0	60,0					
V18_53	62,51	9,82	46,0	89,0					
V18_54	59,52	4,74	47,0	75,0					
V18_55	56,72	8,71	44,0	84,0					
V18_56	57,65	5,78	44,0	66,0					
V18_57	61,52	3,50	50,0	80,0					
V18_58	57,07	10,83	45,0	75,0					
V18_59	70,8	17,67	48,0	95,0					
Média	60,33	7,39	48,0	79,60					

Tab. D.5: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Selecionada com o atributo nível dinâmico e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	53,29	8,41	47,0	83,0	V18_20	48,82	3,99	47,0	60,0
V18_11	47,97	3,68	47,0	67,0	V18_21	60,77	2,71	57,0	78,0
V18_12	56,08	7,24	47,0	69,0	V18_22	57,08	4,19	54,0	79,0
V18_13	47,63	2,71	47,0	62,0	V18_23	49,10	3,92	47,0	71,0
V18_14	47,67	2,73	47,0	62,0	V18_24	46,60	4,09	45,0	64,0
V18_15	47,67	2,73	47,0	62,0	V18_25	58,79	4,77	47,0	82,0
V18_16	47,67	2,73	47,0	62,0	V18_26	49,55	3,66	47,0	62,0
V18_17	50,32	7,81	43,0	66,0	V18_27	44,67	4,07	43,0	62,0
V18_18	47,93	4,32	47,0	82,0	V18_28	47,89	3,44	45,0	67,0
V18_19	47,67	2,73	47,0	62,0	V18_29	54,39	3,26	54,0	86,0
Média	49,39	4,50	46,60	67,70	Média	51,76	3,81	48,60	71,10
V18_30	50,69	9,33	43,0	79,0	V18_40	50,67	9,22	43,0	79,0
V18_31	49,64	7,81	43,0	66,0	V18_41	58,19	3,48	57,0	71,0
V18_32	59,25	4,34	48,0	65,0	V18_42	50,90	1,50	42,0	59,0
V18_33	49,14	6,53	43,0	74,0	V18_43	50,24	5,29	47,0	67,0
V18_34	53,58	11,22	43,0	78,0	V18_44	49,77	9,97	43,0	77,0
V18_35	55,04	7,83	44,0	86,0	V18_45	60,28	7,10	50,0	82,0
V18_36	51,45	6,95	48,0	79,0	V18_46	53,69	2,41	51,0	65,0
V18_37	49,92	4,70	47,0	69,0	V18_47	52,62	9,08	47,0	83,0
V18_38	49,66	5,29	45,0	75,0	V18_48	59,11	0,77	59,0	65,0
V18_39	53,99	6,24	46,0	72,0	V18_49	54,57	3,85	53,0	69,0
Média	52,23	7,02	45,0	74,30	Média	54,01	5,26	49,20	71,70
V18_50	56,52	9,11	49,0	81,0	Média geral	52,18	5,16	47,54	71,40
V18_51	49,20	4,14	47,0	71,0					
V18_52	52,27	4,93	49,0	71,0					
V18_53	63,98	3,78	51,0	69,0					
V18_54	49,73	5,25	47,0	73,0					
V18_55	50,64	7,54	45,0	76,0					
V18_56	52,94	0,69	52,0	58,0					
V18_57	52,31	6,46	48,0	80,0					
V18_58	57,61	7,51	47,0	75,0					
V18_59	50,07	2,97	48,0	68,0					
Média	53,52	5,23	48,30	72,20					

Tab. D.6: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Selecionada com o atributo nível dinâmico e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	48,49	1,52	47,0	54,0	V18_20	48,23	2,79	47,0	60,0
V18_11	47,47	2,02	47,0	60,0	V18_21	56,89	4,95	51,0	70,0
V18_12	47,21	1,22	47,0	56,0	V18_22	47,15	1,50	47,0	62,0
V18_13	47,15	1,07	47,0	56,0	V18_23	48,64	3,39	47,0	58,0
V18_14	47,14	0,81	47,0	53,0	V18_24	45,27	1,65	45,0	58,0
V18_15	47,14	0,81	47,0	53,0	V18_25	47,58	1,65	47,0	56,0
V18_16	47,14	0,81	47,0	53,0	V18_26	47,38	1,48	47,0	56,0
V18_17	45,44	2,26	43,0	54,0	V18_27	43,31	1,69	43,0	56,0
V18_18	47,07	0,69	47,0	54,0	V18_28	48,67	3,90	45,0	61,0
V18_19	47,14	0,81	47,0	53,0	V18_29	51,26	2,82	47,0	55,0
Média	47,13	1,20	46,60	54,60	Média	48,43	2,58	46,60	59,20
V18_30	43,46	1,86	43,0	54,0	V18_40	43,76	2,08	43,0	53,0
V18_31	45,36	1,95	43,0	54,0	V18_41	53,52	1,61	53,0	60,0
V18_32	59,52	10,19	47,0	77,0	V18_42	44,45	4,54	41,0	62,0
V18_33	48,14	2,95	47,0	62,0	V18_43	49,62	2,32	47,0	53,0
V18_34	46,7	1,38	43,0	52,0	V18_44	51,47	4,07	48,0	68,0
V18_35	45,51	2,19	44,0	53,0	V18_45	59,12	8,73	47,0	87,0
V18_36	48,24	1,50	48,0	59,0	V18_46	53,07	2,65	47,0	64,0
V18_37	47,69	1,73	47,0	54,0	V18_47	53,28	3,79	52,0	68,0
V18_38	46,58	3,42	45,0	64,0	V18_48	52,91	2,17	52,0	62,0
V18_39	50,68	6,73	47,0	72,0	V18_49	54,27	2,48	53,0	60,0
Média	48,18	3,39	45,40	60,10	Média	51,54	3,44	48,30	63,70
V18_50	49,2	2,00	49,0	69,0	Média geral	49,32	2,88	46,64	60,04
V18_51	48,05	2,69	47,0	63,0					
V18_52	49,36	3,34	43,0	60,0					
V18_53	63,07	9,53	46,0	77,0					
V18_54	60,07	6,90	50,0	72,0					
V18_55	49,49	2,81	45,0	55,0					
V18_56	47,43	1,76	46,0	54,0					
V18_57	50,74	2,69	48,0	61,0					
V18_58	45,61	3,09	43,0	56,0					
V18_59	50,24	3,57	46,0	59,0					
Média	51,32	3,83	46,30	62,60					

Tab. D.7: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com o atributo nível dinâmico e o modo de operação seqüencial de AC.

D.1.3 Vizinhança Totalística e três atributos

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	51,0	2,88	44,0	59,0	V18_20	48,01	4,97	47,0	87,0
V18_11	53,8	3,40	47,0	66,0	V18_21	49,41	4,71	44,0	67,0
V18_12	51,47	1,97	47,0	53,0	V18_22	52,73	4,62	47,0	63,0
V18_13	48,37	2,56	44,0	55,0	V18_23	51,91	3,44	47,0	61,0
V18_14	48,37	2,56	44,0	55,0	V18_24	49,91	3,33	47,0	77,0
V18_15	53,34	4,45	47,0	60,0	V18_25	50,82	4,76	47,0	79,0
V18_16	54,78	5,16	47,0	68,0	V18_26	49,98	4,36	47,0	71,0
V18_17	46,77	3,07	43,0	56,0	V18_27	48,27	2,74	47,0	57,0
V18_18	48,66	3,63	47,0	62,0	V18_28	48,17	3,53	44,0	60,0
V18_19	48,16	2,79	47,0	59,0	V18_29	52,48	5,12	47,0	67,0
Média	50,47	3,25	45,70	59,30	Média	50,17	4,16	46,40	68,90
V18_30	53,42	0,49	53,0	54,0	V18_40	53,94	3,47	48,0	62,0
V18_31	47,21	3,22	43,0	56,0	V18_41	55,48	3,73	53,0	70,0
V18_32	49,07	2,77	47,0	64,0	V18_42	48,32	4,36	43,0	55,0
V18_33	47,29	2,80	45,0	66,0	V18_43	51,51	3,83	47,0	69,0
V18_34	51,0	0,0	51,0	51,0	V18_44	48,9	3,35	43,0	55,0
V18_35	50,65	8,87	48,0	84,0	V18_45	60,08	7,45	53,0	75,0
V18_36	53,91	3,61	51,0	79,0	V18_46	58,97	3,80	54,0	66,0
V18_37	49,92	3,06	47,0	59,0	V18_47	54,08	3,44	47,0	64,0
V18_38	49,96	3,66	44,0	56,0	V18_48	55,8	2,11	52,0	65,0
V18_39	52,4	2,77	44,0	55,0	V18_49	55,26	3,60	49,0	68,0
Média	50,48	3,13	47,30	62,40	Média	54,23	3,91	48,90	64,90
V18_50	58,19	3,17	49,0	69,0	Média geral	51,97	3,54	47,68	63,90
V18_51	50,32	1,89	49,0	53,0					
V18_52	45,06	0,60	45,0	51,0					
V18_53	56,37	5,17	51,0	74,0					
V18_54	54,75	4,01	49,0	60,0					
V18_55	53,9	4,39	49,0	71,0					
V18_56	62,5	4,91	57,0	79,0					
V18_57	51,87	3,37	47,0	63,0					
V18_58	55,63	2,00	52,0	58,0					
V18_59	56,72	3,14	53,0	62,0					
Média	54,53	3,27	50,10	64,0					

Tab. D.8: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhança Totalística com três atributos diferentes e o modo de operação seqüencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,88	3,29	44,0	62,0	V18_20	47,98	2,02	47,0	53,0
V18_11	52,17	2,57	47,0	57,0	V18_21	50,62	1,93	44,0	53,0
V18_12	48,88	1,68	44,0	53,0	V18_22	52,36	3,57	47,0	57,0
V18_13	48,53	2,73	44,0	55,0	V18_23	50,19	2,21	49,0	55,0
V18_14	48,53	2,73	44,0	55,0	V18_24	51,31	3,88	45,0	57,0
V18_15	53,22	3,60	47,0	64,0	V18_25	48,47	2,24	47,0	53,0
V18_16	53,45	5,92	47,0	59,0	V18_26	49,29	2,25	47,0	55,0
V18_17	46,28	1,99	44,0	48,0	V18_27	48,53	2,73	44,0	55,0
V18_18	49,86	3,06	44,0	55,0	V18_28	48,36	2,30	47,0	53,0
V18_19	48,62	2,74	44,0	55,0	V18_29	49,99	2,85	47,0	63,0
Média	49,94	3,03	44,90	56,30	Média	49,71	2,60	46,40	55,40
V18_30	54,49	4,58	50,0	66,0	V18_40	56,56	5,06	49,0	67,0
V18_31	46,28	1,99	44,0	48,0	V18_41	54,0	2,42	53,0	71,0
V18_32	49,13	3,57	47,0	65,0	V18_42	43,76	0,65	42,0	44,0
V18_33	49,78	2,35	47,0	53,0	V18_43	49,79	2,99	47,0	69,0
V18_34	51,0	0,0	51,0	51,0	V18_44	47,2	2,47	44,0	52,0
V18_35	47,17	2,06	44,0	59,0	V18_45	56,66	5,91	46,0	70,0
V18_36	53,82	1,47	47,0	60,0	V18_46	53,55	6,04	47,0	60,0
V18_37	50,84	1,74	47,0	53,0	V18_47	47,92	1,00	47,0	49,0
V18_38	51,46	2,10	48,0	53,0	V18_48	53,98	1,52	52,0	57,0
V18_39	50,57	2,89	44,0	55,0	V18_49	54,02	2,08	50,0	57,0
Média	50,45	2,28	46,90	56,30	Média	51,74	3,01	47,70	59,60
V18_50	54,19	1,90	49,0	60,0	Média geral	50,92	2,74	46,78	57,72
V18_51	50,64	1,97	49,0	53,0					
V18_52	48,66	2,67	42,0	58,0					
V18_53	51,7	4,15	46,0	67,0					
V18_54	51,4	1,96	49,0	53,0					
V18_55	50,15	2,90	48,0	71,0					
V18_56	55,32	4,30	49,0	65,0					
V18_57	52,55	2,47	47,0	58,0					
V18_58	58,9	4,54	54,0	70,0					
V18_59	54,34	1,35	47,0	55,0					
Média	52,79	2,82	48,0	61,0					

Tab. D.9: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Totalística com três atributos diferentes e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,69	2,49	44,0	55,0	V18_20	49,63	2,36	44,0	53,0
V18_11	53,64	2,87	48,0	62,0	V18_21	50,24	2,96	44,0	65,0
V18_12	50,81	2,87	44,0	61,0	V18_22	51,81	3,12	44,0	65,0
V18_13	49,49	2,51	44,0	53,0	V18_23	52,44	1,92	47,0	57,0
V18_14	49,49	2,51	44,0	53,0	V18_24	51,37	3,78	45,0	67,0
V18_15	50,95	3,84	44,0	65,0	V18_25	50,22	1,87	47,0	55,0
V18_16	52,64	4,47	44,0	65,0	V18_26	50,47	3,28	47,0	55,0
V18_17	44,0	0,0	44,0	44,0	V18_27	49,49	2,51	44,0	53,0
V18_18	50,02	2,58	44,0	59,0	V18_28	49,63	2,44	44,0	53,0
V18_19	49,54	2,38	44,0	53,0	V18_29	52,25	1,89	47,0	58,0
Média	50,02	2,65	44,40	57,0	Média	50,75	2,61	45,30	58,10
V18_30	53,0	3,11	48,0	70,0	V18_40	53,73	3,47	47,0	67,0
V18_31	44,0	0,0	44,0	44,0	V18_41	56,24	1,70	53,0	58,0
V18_32	49,87	1,88	47,0	55,0	V18_42	40,75	2,46	40,0	49,0
V18_33	46,88	2,00	45,0	49,0	V18_43	52,13	3,69	47,0	69,0
V18_34	50,95	1,16	47,0	56,0	V18_44	48,43	3,11	44,0	57,0
V18_35	46,62	1,86	44,0	48,0	V18_45	55,59	4,69	46,0	64,0
V18_36	56,48	3,52	50,0	76,0	V18_46	51,52	2,99	44,0	55,0
V18_37	50,44	2,92	47,0	67,0	V18_47	50,84	1,25	47,0	53,0
V18_38	52,15	4,32	44,0	58,0	V18_48	55,73	2,48	49,0	59,0
V18_39	52,52	3,57	44,0	62,0	V18_49	52,54	2,63	50,0	57,0
Média	50,29	2,43	46,0	58,50	Média	51,75	2,84	46,70	58,80
V18_50	54,0	0,0	54,0	54,0	Média geral	51,07	2,57	46,26	58,0
V18_51	51,2	1,99	49,0	53,0					
V18_52	47,0	0,0	47,0	47,0					
V18_53	51,97	0,65	51,0	56,0					
V18_54	54,21	4,84	49,0	67,0					
V18_55	50,94	4,35	45,0	60,0					
V18_56	50,7	1,94	47,0	55,0					
V18_57	55,7	3,54	46,0	58,0					
V18_58	55,01	1,86	52,0	58,0					
V18_59	54,96	4,08	49,0	68,0					
Média	52,56	2,32	48,90	57,60					

Tab. D.10: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Totalística com três atributos diferentes e o modo de operação sequencial de AC.

D.1.4 Vizinhaça Totalística e o atributo nível dinâmico

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	45,57	2,10	44,0	54,0	V18_20	47,47	2,77	43,0	55,0
V18_11	47,81	2,25	47,0	57,0	V18_21	45,26	1,85	44,0	53,0
V18_12	45,11	1,78	44,0	52,0	V18_22	44,79	1,59	44,0	56,0
V18_13	44,68	1,94	44,0	53,0	V18_23	48,97	2,10	47,0	54,0
V18_14	44,59	1,90	44,0	53,0	V18_24	46,14	2,10	45,0	55,0
V18_15	44,59	1,90	44,0	53,0	V18_25	48,73	1,89	47,0	53,0
V18_16	44,59	1,90	44,0	53,0	V18_26	48,43	2,88	44,0	55,0
V18_17	43,97	1,44	43,0	52,0	V18_27	44,3	2,18	43,0	55,0
V18_18	45,8	1,47	44,0	47,0	V18_28	44,62	2,00	44,0	55,0
V18_19	44,5	1,70	44,0	53,0	V18_29	49,22	3,99	47,0	61,0
Média	45,12	1,84	44,20	52,70	Média	46,79	2,34	44,80	55,20
V18_30	55,83	4,56	49,0	62,0	V18_40	56,41	4,52	49,0	62,0
V18_31	43,22	0,41	43,0	44,0	V18_41	53,87	2,25	53,0	65,0
V18_32	50,24	7,32	47,0	92,0	V18_42	43,25	2,85	40,0	48,0
V18_33	46,59	1,67	44,0	53,0	V18_43	46,62	2,13	45,0	53,0
V18_34	47,21	3,90	43,0	62,0	V18_44	45,41	2,49	44,0	53,0
V18_35	48,0	0,0	48,0	48,0	V18_45	47,22	1,22	45,0	51,0
V18_36	51,41	4,39	48,0	66,0	V18_46	49,67	3,20	44,0	66,0
V18_37	45,63	2,95	44,0	53,0	V18_47	48,68	3,16	47,0	62,0
V18_38	50,54	3,38	44,0	64,0	V18_48	49,45	2,80	48,0	57,0
V18_39	48,35	3,03	47,0	62,0	V18_49	54,77	2,85	50,0	59,0
Média	48,70	3,16	45,70	60,60	Média	49,54	2,75	46,50	57,60
V18_50	51,55	3,00	49,0	65,0	Média geral	48,15	2,60	45,78	57,46
V18_51	49,05	0,50	49,0	54,0					
V18_52	47,29	2,74	44,0	53,0					
V18_53	54,94	6,64	51,0	79,0					
V18_54	49,01	2,44	47,0	66,0					
V18_55	48,73	3,06	45,0	55,0					
V18_56	47,83	3,97	44,0	61,0					
V18_57	49,81	2,76	47,0	58,0					
V18_58	54,39	2,50	52,0	59,0					
V18_59	53,36	1,77	49,0	62,0					
Média	50,60	2,94	47,70	61,20					

Tab. D.11: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	50,32	1,58	47,0	53,0	V18_20	48,49	2,13	47,0	53,0
V18_11	53,9	2,97	47,0	63,0	V18_21	48,67	2,02	47,0	52,0
V18_12	50,16	1,92	47,0	53,0	V18_22	50,24	2,79	47,0	59,0
V18_13	48,45	1,94	47,0	52,0	V18_23	51,74	2,64	47,0	57,0
V18_14	48,51	2,11	47,0	53,0	V18_24	50,27	2,59	45,0	55,0
V18_15	48,51	2,11	47,0	53,0	V18_25	47,29	1,06	47,0	52,0
V18_16	48,51	2,11	47,0	53,0	V18_26	50,88	3,38	47,0	55,0
V18_17	45,4	1,91	44,0	48,0	V18_27	48,37	2,15	43,0	52,0
V18_18	50,2	2,62	44,0	57,0	V18_28	48,53	2,30	45,0	52,0
V18_19	48,51	2,11	47,0	53,0	V18_29	52,96	2,23	47,0	55,0
Média	49,25	2,14	46,40	53,80	Média	49,74	2,33	46,20	54,20
V18_30	52,45	2,34	47,0	59,0	V18_40	52,74	2,00	47,0	59,0
V18_31	44,05	0,82	43,0	52,0	V18_41	56,68	0,97	53,0	57,0
V18_32	50,08	1,65	47,0	51,0	V18_42	44,81	0,76	41,0	45,0
V18_33	51,63	3,04	45,0	65,0	V18_43	50,69	2,33	47,0	60,0
V18_34	50,55	0,50	50,0	51,0	V18_44	49,24	1,85	48,0	52,0
V18_35	48,73	2,07	48,0	56,0	V18_45	52,64	3,07	47,0	55,0
V18_36	53,69	3,74	48,0	67,0	V18_46	53,35	2,53	44,0	55,0
V18_37	50,22	1,70	47,0	53,0	V18_47	57,9	3,27	51,0	68,0
V18_38	52,22	3,56	44,0	57,0	V18_48	55,91	3,80	49,0	75,0
V18_39	53,73	3,83	47,0	71,0	V18_49	57,63	2,61	54,0	65,0
Média	50,74	2,33	46,60	58,20	Média	53,16	2,32	48,10	59,10
V18_50	51,73	2,10	49,0	58,0	Média geral	51,08	2,31	47,0	56,46
V18_51	52,14	2,42	46,0	57,0					
V18_52	48,76	0,95	45,0	49,0					
V18_53	53,15	2,37	49,0	56,0					
V18_54	50,54	2,73	47,0	59,0					
V18_55	51,67	2,84	46,0	58,0					
V18_56	50,48	1,54	44,0	52,0					
V18_57	55,4	3,29	50,0	58,0					
V18_58	57,39	3,87	49,0	61,0					
V18_59	53,8	2,28	52,0	62,0					
Média	52,51	2,44	47,70	57,0					

Tab. D.12: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	46,17	3,04	44,0	54,0	V18_20	47,96	3,67	43,0	55,0
V18_11	49,06	3,09	47,0	59,0	V18_21	46,78	3,03	44,0	52,0
V18_12	46,51	2,85	44,0	53,0	V18_22	45,96	2,55	44,0	52,0
V18_13	45,54	2,68	44,0	52,0	V18_23	50,96	2,81	47,0	57,0
V18_14	45,45	2,58	44,0	52,0	V18_24	47,32	2,72	45,0	54,0
V18_15	45,45	2,58	44,0	52,0	V18_25	47,91	1,92	47,0	53,0
V18_16	45,45	2,58	44,0	52,0	V18_26	50,14	3,21	47,0	62,0
V18_17	44,96	1,71	44,0	48,0	V18_27	45,57	2,78	43,0	53,0
V18_18	47,7	3,67	44,0	54,0	V18_28	45,46	2,62	44,0	52,0
V18_19	45,49	2,63	44,0	52,0	V18_29	51,29	2,98	47,0	55,0
Média	46,17	2,74	44,30	52,80	Média	47,93	2,82	45,10	54,50
V18_30	52,24	1,98	49,0	54,0	V18_40	53,42	2,16	49,0	59,0
V18_31	45,0	1,74	44,0	48,0	V18_41	55,53	3,03	53,0	67,0
V18_32	48,47	1,67	47,0	52,0	V18_42	43,62	2,68	41,0	49,0
V18_33	50,52	2,70	46,0	55,0	V18_43	49,42	2,92	45,0	64,0
V18_34	50,75	0,55	50,0	52,0	V18_44	48,92	1,69	48,0	52,0
V18_35	48,0	0,0	48,0	48,0	V18_45	48,71	2,78	47,0	55,0
V18_36	53,03	4,44	48,0	67,0	V18_46	51,57	2,65	48,0	55,0
V18_37	49,06	2,56	44,0	55,0	V18_47	52,29	5,59	47,0	83,0
V18_38	52,3	3,54	46,0	59,0	V18_48	51,59	2,83	48,0	57,0
V18_39	48,13	2,35	44,0	55,0	V18_49	55,65	4,12	50,0	61,0
Média	49,75	2,15	46,60	54,50	Média	51,07	3,04	47,60	60,20
V18_50	51,99	2,34	49,0	58,0	Média geral	49,27	2,76	46,36	56,52
V18_51	50,07	2,39	46,0	57,0					
V18_52	47,08	2,30	45,0	53,0					
V18_53	52,0	0,0	52,0	52,0					
V18_54	48,69	3,04	47,0	71,0					
V18_55	57,89	9,95	48,0	84,0					
V18_56	48,17	3,26	44,0	55,0					
V18_57	52,2	3,81	47,0	58,0					
V18_58	52,72	0,89	52,0	56,0					
V18_59	53,85	2,65	52,0	62,0					
Média	51,46	3,06	48,20	60,60					

Tab. D.13: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_60	45,08	1,70	43,0	49,0
V18_61	44,09	2,68	41,0	49,0
V18_62	47,82	5,27	45,0	65,0
V18_63	51,64	2,40	49,0	57,0
V18_64	56,45	5,20	50,0	68,0
V18_65	51,08	2,26	50,0	58,0
V18_66	47,41	4,08	44,0	59,0
V18_67	42,71	4,50	38,0	50,0
V18_68	47,55	3,08	45,0	60,0
V18_69	47,6	3,34	44,0	57,0
Média	48,14	3,45	44,90	57,20
V18_70	48,29	0,94	46,0	51,0
V18_71	48,21	0,98	48,0	54,0
V18_72	60,7	7,52	55,0	79,0
V18_73	45,37	1,22	44,0	50,0
V18_74	47,72	5,08	44,0	62,0
V18_75	50,87	3,46	48,0	55,0
V18_76	47,54	3,42	44,0	56,0
V18_77	47,54	3,66	40,0	54,0
V18_78	45,42	3,19	42,0	54,0
V18_79	50,91	4,20	47,0	58,0
Média	49,25	3,36	45,80	57,30
V18_80	46,6	3,06	42,0	58,0
V18_81	52,53	5,90	48,0	64,0
V18_82	58,96	2,10	53,0	61,0
V18_83	49,48	1,64	49,0	57,0
V18_84	44,64	2,93	42,0	58,0
V18_85	45,88	2,48	42,0	55,0
V18_86	38,02	1,63	37,0	41,0
V18_87	52,59	5,34	45,0	61,0
V18_88	58,28	8,73	47,0	90,0
V18_89	47,24	5,74	43,0	66,0
Média	49,42	3,95	44,80	61,10
Média Geral	48,93	3,58	45,16	58,53

Tab. D.14: Valores obtidos nas 30 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação sequencial de AC.

D.2 Modo de operação paralelo de AC

D.2.1 Vizinhaça Selecionada e três atributos

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	53,61	13,33	44,0	87,0	V18_20	46,8	6,57	44,0	69,0
V18_11	64,62	9,01	50,0	78,0	V18_21	57,8	3,68	53,0	69,0
V18_12	60,0	8,91	53,0	85,0	V18_22	57,16	8,65	46,0	97,0
V18_13	49,19	10,32	44,0	95,0	V18_23	58,57	7,53	49,0	84,0
V18_14	44,0	0,00	44,0	44,0	V18_24	68,39	8,76	51,0	94,0
V18_15	57,14	10,94	47,0	95,0	V18_25	60,31	10,17	53,0	95,0
V18_16	61,07	9,27	53,0	94,0	V18_26	59,65	8,75	48,0	85,0
V18_17	47,81	6,09	46,0	82,0	V18_27	44,0	0,00	44,0	44,0
V18_18	56,92	13,87	44,0	102,0	V18_28	44,69	3,67	44,0	75,0
V18_19	59,89	1,75	53,0	63,0	V18_29	53,33	4,74	44,0	73,0
Média	55,42	8,36	47,80	82,50	Média	55,07	6,25	47,60	78,50
V18_30	51,13	3,81	44,0	69,0	V18_40	59,55	9,66	46,0	75,0
V18_31	47,81	6,09	46,0	82,0	V18_41	65,1	8,60	55,0	85,0
V18_32	67,05	12,36	52,0	89,0	V18_42	53,78	3,00	47,0	65,0
V18_33	51,32	10,35	44,0	79,0	V18_43	59,91	9,09	52,0	76,0
V18_34	50,0	0,00	50,0	50,0	V18_44	60,78	7,40	46,0	78,0
V18_35	48,64	6,64	47,0	83,0	V18_45	64,23	14,22	50,0	95,0
V18_36	64,31	1,63	64,0	76,0	V18_46	63,61	8,71	44,0	90,0
V18_37	53,82	6,91	44,0	82,0	V18_47	57,35	11,17	50,0	84,0
V18_38	60,76	9,08	46,0	82,0	V18_48	60,23	4,19	56,0	79,0
V18_39	56,11	14,73	44,0	92,0	V18_49	64,64	7,01	52,0	87,0
Média	55,09	7,16	48,10	78,40	Média	60,91	8,30	49,80	81,40
V18_50	68,46	7,22	59,0	94,0	Média geral	57,11	7,78	48,50	81,30
V18_51	50,23	11,40	46,0	93,0					
V18_52	53,79	2,68	53,0	72,0					
V18_53	72,04	14,77	48,0	94,0					
V18_54	61,24	9,61	51,0	85,0					
V18_55	54,94	12,51	45,0	88,0					
V18_56	63,38	5,46	52,0	77,0					
V18_57	49,47	1,78	49,0	64,0					
V18_58	50,94	10,17	44,0	82,0					
V18_59	66,46	12,96	45,0	108,0					
Média	59,09	8,85	49,20	85,70					

Tab. D.15: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Selecionada com três atributos diferentes e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,0	0,00	49,0	49,0	V18_20	57,9	3,18	53,0	73,0
V18_11	53,29	2,05	53,0	69,0	V18_21	61,63	9,37	55,0	84,0
V18_12	57,34	9,60	52,0	94,0	V18_22	49,68	3,46	49,0	73,0
V18_13	49,86	3,08	49,0	67,0	V18_23	51,0	0,00	51,0	51,0
V18_14	49,59	2,43	49,0	63,0	V18_24	51,44	3,30	51,0	83,0
V18_15	50,19	2,34	49,0	63,0	V18_25	61,24	9,47	52,0	83,0
V18_16	50,1	4,95	49,0	83,0	V18_26	51,02	7,07	49,0	85,0
V18_17	51,97	5,55	49,0	82,0	V18_27	49,59	2,43	49,0	63,0
V18_18	53,29	6,72	49,0	73,0	V18_28	57,72	7,83	52,0	85,0
V18_19	49,49	1,78	49,0	57,0	V18_29	57,53	11,74	49,0	94,0
Média	51,41	3,85	49,70	70,0	Média	54,87	5,78	51,0	77,4
V18_30	53,0	8,45	49,0	77,0	V18_40	53,03	8,29	49,0	78,0
V18_31	53,11	5,11	49,0	82,0	V18_41	64,31	8,00	57,0	83,0
V18_32	67,3	11,49	55,0	110,0	V18_42	55,9	4,79	51,0	68,0
V18_33	49,82	3,98	45,0	79,0	V18_43	57,86	3,35	56,0	72,0
V18_34	56,79	8,38	49,0	85,0	V18_44	59,03	4,56	55,0	84,0
V18_35	46,42	2,24	46,0	64,0	V18_45	63,74	6,45	47,0	83,0
V18_36	54,22	2,20	54,0	76,0	V18_46	62,17	8,32	49,0	78,0
V18_37	51,39	8,58	49,0	102,0	V18_47	55,63	10,21	49,0	86,0
V18_38	47,53	1,89	47,0	62,0	V18_48	63,96	4,71	62,0	100,0
V18_39	53,76	9,71	49,0	85,0	V18_49	67,67	9,30	54,0	91,0
Média	53,33	6,20	49,20	82,20	Média	60,33	6,79	52,90	82,23
V18_50	62,05	8,07	51,0	80,0	Média geral	56,27	6,23	50,88	79,88
V18_51	66,32	10,06	52,0	95,0					
V18_52	55,44	4,99	53,0	77,0					
V18_53	72,75	14,19	52,0	101,0					
V18_54	66,85	10,67	51,0	92,0					
V18_55	58,91	10,37	48,0	88,0					
V18_56	57,91	9,79	51,0	94,0					
V18_57	55,47	6,82	52,0	89,0					
V18_58	53,88	5,22	51,0	71,0					
V18_59	64,64	5,16	55,0	89,0					
Média	61,42	8,53	51,60	87,60					

Tab. D.16: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	47,85	1,88	47,0	52,0	V18_20	47,82	0,98	47,0	49,0
V18_11	51,18	0,57	51,0	53,0	V18_21	53,76	2,49	53,0	78,0
V18_12	47,0	0,0	47,0	47,0	V18_22	47,87	3,35	47,0	67,0
V18_13	47,23	2,30	47,0	70,0	V18_23	49,04	0,28	49,0	51,0
V18_14	47,34	0,75	47,0	49,0	V18_24	52,28	5,46	47,0	61,0
V18_15	47,7	0,95	47,0	49,0	V18_25	47,06	0,34	47,0	49,0
V18_16	47,18	1,80	47,0	65,0	V18_26	47,12	0,47	47,0	49,0
V18_17	46,38	0,48	46,0	47,0	V18_27	47,34	0,75	47,0	49,0
V18_18	47,56	1,99	47,0	65,0	V18_28	47,0	0,0	47,0	47,0
V18_19	47,16	0,54	47,0	49,0	V18_29	52,4	2,34	49,0	54,0
Média	47,65	1,12	47,30	54,60	Média	49,16	1,64	48,0	55,40
V18_30	47,33	1,58	47,0	62,0	V18_40	47,42	0,81	47,0	49,0
V18_31	46,38	0,48	46,0	47,0	V18_41	53,28	0,69	53,0	55,0
V18_32	48,6	0,92	48,0	50,0	V18_42	46,43	0,49	46,0	47,0
V18_33	51,47	5,74	47,0	69,0	V18_43	49,0	0,0	49,0	49,0
V18_34	46,24	0,42	46,0	47,0	V18_44	49,88	3,76	46,0	65,0
V18_35	50,57	3,04	48,0	61,0	V18_45	49,96	1,65	48,0	53,0
V18_36	52,26	0,67	52,0	54,0	V18_46	47,82	0,53	47,0	49,0
V18_37	47,52	1,41	47,0	53,0	V18_47	54,94	5,22	50,0	74,0
V18_38	49,69	2,23	47,0	53,0	V18_48	53,16	0,54	53,0	55,0
V18_39	47,0	0,0	47,0	47,0	V18_49	53,0	0,0	53,0	53,0
Média	48,70	1,64	47,50	54,30	Média	50,48	1,36	49,20	54,90
V18_50	56,66	5,06	53,0	77,0	Média geral	49,71	1,78	48,30	56,60
V18_51	49,24	0,65	49,0	51,0					
V18_52	47,67	4,23	47,0	84,0					
V18_53	53,54	3,29	51,0	68,0					
V18_54	49,0	0,0	49,0	49,0					
V18_55	51,27	4,52	47,0	64,0					
V18_56	50,94	3,07	47,0	54,0					
V18_57	53,48	0,85	53,0	55,0					
V18_58	52,03	2,78	47,0	61,0					
V18_59	61,93	6,97	52,0	75,0					
Média	52,57	3,14	49,50	63,80					

Tab. D.17: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhaça Seleccionada com três atributos diferentes e o modo de operação paralelo de AC.

D.2.2 Vizinhaça Seleccionada e o atributo nível dinâmico

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	65,93	8,30	55,0	94,0	V18_20	59,91	8,74	51,0	90,0
V18_11	59,77	9,65	48,0	99,0	V18_21	58,32	5,55	53,0	85,0
V18_12	60,18	10,18	51,0	91,0	V18_22	53,96	12,37	44,0	86,0
V18_13	54,23	6,84	44,0	91,0	V18_23	61,94	8,61	52,0	85,0
V18_14	55,4	7,63	44,0	82,0	V18_24	57,64	7,14	47,0	82,0
V18_15	55,4	7,63	44,0	82,0	V18_25	65,92	10,39	53,0	95,0
V18_16	55,4	7,63	44,0	82,0	V18_26	56,11	9,81	44,0	95,0
V18_17	51,9	7,52	46,0	90,0	V18_27	55,48	8,90	44,0	95,0
V18_18	58,6	7,01	44,0	85,0	V18_28	52,58	7,43	44,0	94,0
V18_19	55,4	7,63	44,0	82,0	V18_29	72,44	11,13	44,0	95,0
Média	57,22	8,00	46,40	87,80	Média	59,43	9,00	47,60	90,20
V18_30	53,41	5,99	47,0	80,0	V18_40	53,48	5,95	47,0	80,0
V18_31	58,28	7,40	48,0	75,0	V18_41	61,75	8,39	56,0	95,0
V18_32	58,07	12,24	47,0	102,0	V18_42	53,38	4,46	47,0	69,0
V18_33	54,26	11,19	44,0	86,0	V18_43	59,28	7,82	50,0	88,0
V18_34	50,96	3,55	47,0	58,0	V18_44	63,65	15,46	44,0	97,0
V18_35	48,05	0,72	47,0	55,0	V18_45	59,74	11,01	50,0	83,0
V18_36	58,67	7,55	56,0	100,0	V18_46	59,02	6,51	54,0	82,0
V18_37	56,77	8,54	53,0	93,0	V18_47	63,67	7,30	53,0	75,0
V18_38	63,29	5,52	51,0	74,0	V18_48	58,36	6,11	49,0	99,0
V18_39	68,91	9,84	52,0	92,0	V18_49	66,27	6,15	57,0	86,0
Média	57,06	7,25	49,20	81,50	Média	59,86	7,91	50,70	85,40
V18_50	76,82	10,03	57,0	97,0	Média geral	58,94	8,17	49,12	87,14
V18_51	57,18	8,60	53,0	89,0					
V18_52	53,1	4,06	48,0	71,0					
V18_53	57,61	11,40	51,0	90,0					
V18_54	57,55	8,55	53,0	95,0					
V18_55	64,09	12,07	51,0	108,0					
V18_56	62,08	6,69	55,0	91,0					
V18_57	58,65	8,99	46,0	84,0					
V18_58	59,18	6,99	49,0	83,0					
V18_59	65,19	9,99	54,0	100,0					
Média	61,14	8,73	51,70	90,80					

Tab. D.18: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	52,93	7,29	49,0	87,0	V18_20	47,28	1,02	47,0	51,0
V18_11	47,68	2,27	47,0	59,0	V18_21	54,54	4,85	52,0	77,0
V18_12	50,14	4,25	47,0	82,0	V18_22	56,75	2,64	51,0	66,0
V18_13	47,2	0,87	47,0	51,0	V18_23	47,44	1,28	47,0	53,0
V18_14	47,2	0,87	47,0	51,0	V18_24	45,48	1,82	45,0	53,0
V18_15	47,2	0,87	47,0	51,0	V18_25	57,73	6,98	53,0	85,0
V18_16	47,2	0,87	47,0	51,0	V18_26	47,85	3,11	47,0	65,0
V18_17	50,6	7,64	46,0	83,0	V18_27	43,52	1,94	43,0	51,0
V18_18	47,2	0,87	47,0	51,0	V18_28	45,48	1,48	45,0	51,0
V18_19	47,2	0,87	47,0	51,0	V18_29	64,77	2,99	55,0	70,0
Média	48,45	2,66	47,10	61,70	Média	51,08	2,81	48,45	62,20
V18_30	43,48	1,50	43,0	53,0	V18_40	43,51	1,48	43,0	52,0
V18_31	51,87	8,91	46,0	83,0	V18_41	54,12	1,80	53,0	57,0
V18_32	49,78	5,56	48,0	76,0	V18_42	54,0	0,56	50,0	58,0
V18_33	47,61	2,71	47,0	72,0	V18_43	55,36	8,81	47,0	84,0
V18_34	43,13	0,91	43,0	52,0	V18_44	60,48	12,08	47,0	84,0
V18_35	47,48	1,35	44,0	48,0	V18_45	54,12	8,08	49,0	86,0
V18_36	49,28	2,28	48,0	56,0	V18_46	51,0	0,0	51,0	51,0
V18_37	48,17	2,65	47,0	56,0	V18_47	56,89	4,85	51,0	76,0
V18_38	57,06	4,10	51,0	64,0	V18_48	60,3	3,07	53,0	67,0
V18_39	49,36	5,21	47,0	65,0	V18_49	53,43	1,21	53,0	57,0
Média	48,72	3,51	46,40	62,50	Média	54,32	4,19	49,70	67,20
V18_50	56,35	10,35	49,0	79,0	Média geral	51,04	3,61	47,91	64,98
V18_51	47,4	1,36	47,0	53,0					
V18_52	55,87	5,99	47,0	83,0					
V18_53	47,08	4,38	46,0	76,0					
V18_54	48,02	3,75	47,0	72,0					
V18_55	55,71	5,39	48,0	71,0					
V18_56	46,94	1,30	46,0	55,0					
V18_57	56,43	7,99	48,0	78,0					
V18_58	59,18	3,10	53,0	75,0					
V18_59	53,37	5,28	48,0	71,0					
Média	52,63	4,88	47,90	71,30					

Tab. D.19: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	49,43	1,90	47,0	53,0	V18_20	47,29	0,87	47,0	51,0
V18_11	51,24	0,95	51,0	55,0	V18_21	49,15	3,88	47,0	67,0
V18_12	48,57	2,18	47,0	57,0	V18_22	48,04	1,58	47,0	51,0
V18_13	47,4	1,30	47,0	57,0	V18_23	51,0	0,0	51,0	51,0
V18_14	47,42	1,34	47,0	57,0	V18_24	49,36	1,00	49,0	53,0
V18_15	47,42	1,34	47,0	57,0	V18_25	47,48	3,52	47,0	78,0
V18_16	47,42	1,34	47,0	57,0	V18_26	47,16	0,78	47,0	51,0
V18_17	47,17	0,58	47,0	50,0	V18_27	47,42	1,34	47,0	57,0
V18_18	47,29	1,15	47,0	57,0	V18_28	47,2	0,87	47,0	51,0
V18_19	47,42	1,34	47,0	57,0	V18_29	50,35	2,16	49,0	59,0
Média	48,07	1,34	47,40	55,70	Média	48,44	1,60	47,80	56,90
V18_30	50,14	2,13	47,0	63,0	V18_40	50,14	2,13	47,0	63,0
V18_31	47,23	0,86	47,0	52,0	V18_41	55,0	0,0	55,0	55,0
V18_32	50,39	5,41	47,0	76,0	V18_42	47,09	2,89	43,0	54,0
V18_33	45,0	0,0	45,0	45,0	V18_43	49,95	0,21	49,0	50,0
V18_34	47,2	1,00	47,0	53,0	V18_44	48,27	0,98	47,0	50,0
V18_35	48,67	1,65	47,0	52,0	V18_45	55,64	1,88	53,0	67,0
V18_36	54,0	0,0	54,0	54,0	V18_46	47,22	1,54	47,0	58,0
V18_37	47,4	1,36	47,0	52,0	V18_47	51,44	3,99	48,0	75,0
V18_38	50,89	4,92	45,0	63,0	V18_48	52,31	1,16	52,0	57,0
V18_39	50,0	0,0	50,0	50,0	V18_49	55,95	0,47	53,0	58,0
Média	49,09	1,73	47,60	56,0	Média	51,30	1,52	49,40	58,70
V18_50	57,45	1,67	53,0	59,0	Média geral	49,69	1,72	48,24	57,76
V18_51	50,14	2,29	49,0	59,0					
V18_52	47,15	1,97	43,0	59,0					
V18_53	51,27	6,92	47,0	71,0					
V18_54	50,54	2,65	49,0	67,0					
V18_55	50,64	1,97	49,0	53,0					
V18_56	49,26	2,51	47,0	64,0					
V18_57	54,56	3,42	52,0	71,0					
V18_58	52,02	0,28	50,0	53,0					
V18_59	52,84	0,77	51,0	59,0					
Média	51,58	2,44	49,0	61,50					

Tab. D.20: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Seleccionada com atributo nível dinâmico e o modo de operação paralelo de AC.

D.2.3 Vizinhança Totalística e três atributos

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	53,1	1,95	50,0	58,0	V18_20	51,36	3,02	47,0	58,0
V18_11	55,56	3,27	51,0	62,0	V18_21	53,01	2,75	50,0	70,0
V18_12	51,66	2,35	49,0	58,0	V18_22	53,76	4,72	47,0	75,0
V18_13	51,76	3,35	47,0	58,0	V18_23	53,04	3,52	49,0	60,0
V18_14	51,76	3,35	47,0	58,0	V18_24	51,47	3,55	47,0	62,0
V18_15	51,65	3,36	47,0	58,0	V18_25	52,3	2,82	47,0	58,0
V18_16	52,64	1,47	52,0	56,0	V18_26	51,53	3,26	47,0	58,0
V18_17	53,97	6,49	46,0	83,0	V18_27	51,76	3,35	47,0	58,0
V18_18	55,32	3,60	51,0	78,0	V18_28	51,4	2,93	47,0	58,0
V18_19	51,75	3,27	47,0	58,0	V18_29	55,34	2,69	49,0	61,0
Média	52,91	3,24	48,70	62,70	Média	52,49	3,26	47,70	61,80
V18_30	51,78	3,42	47,0	58,0	V18_40	51,78	3,42	47,0	58,0
V18_31	54,02	3,44	50,0	72,0	V18_41	57,1	2,63	53,0	69,0
V18_32	53,95	5,23	48,0	64,0	V18_42	53,34	7,11	47,0	67,0
V18_33	55,88	5,18	53,0	69,0	V18_43	54,11	3,98	49,0	61,0
V18_34	49,36	1,15	49,0	53,0	V18_44	57,49	7,75	52,0	85,0
V18_35	52,9	2,69	48,0	59,0	V18_45	54,75	1,17	54,0	59,0
V18_36	57,28	3,00	54,0	64,0	V18_46	54,65	5,78	47,0	82,0
V18_37	54,85	5,80	47,0	71,0	V18_47	52,88	1,67	52,0	56,0
V18_38	50,44	2,66	47,0	60,0	V18_48	57,57	3,07	53,0	64,0
V18_39	56,59	7,28	47,0	95,0	V18_49	58,4	1,21	58,0	62,0
Média	53,70	3,98	49,0	66,50	Média	55,20	3,77	51,20	66,30
V18_50	64,59	4,18	54,0	72,0	Média geral	54,03	3,43	49,60	64,90
V18_51	58,13	3,37	53,0	77,0					
V18_52	53,47	3,82	47,0	71,0					
V18_53	56,0	0,00	56,0	56,0					
V18_54	56,25	4,06	52,0	75,0					
V18_55	53,91	4,86	48,0	80,0					
V18_56	49,47	2,48	47,0	58,0					
V18_57	57,91	2,51	53,0	64,0					
V18_58	51,62	2,37	47,0	58,0					
V18_59	57,56	1,39	57,0	61,0					
Média	55,89	2,90	51,40	67,20					

Tab. D.21: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhança Totalística com três atributos diferentes e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	52,25	3,20	47,0	64,0	V18_20	51,17	1,70	47,0	59,0
V18_11	57,02	2,59	51,0	80,0	V18_21	54,15	1,87	49,0	61,0
V18_12	54,66	3,97	47,0	66,0	V18_22	53,29	1,63	50,0	64,0
V18_13	52,23	1,77	47,0	55,0	V18_23	55,18	3,66	47,0	65,0
V18_14	52,23	1,77	47,0	55,0	V18_24	54,93	1,05	51,0	59,0
V18_15	55,91	3,90	49,0	65,0	V18_25	54,73	4,57	50,0	73,0
V18_16	55,52	4,40	47,0	66,0	V18_26	52,02	3,82	47,0	77,0
V18_17	54,38	4,95	46,0	71,0	V18_27	51,88	2,65	49,0	67,0
V18_18	50,84	3,80	47,0	71,0	V18_28	50,29	3,62	44,0	77,0
V18_19	51,58	2,90	47,0	77,0	V18_29	53,31	3,65	51,0	69,0
Média	53,66	3,32	47,50	67,0	Média	53,09	2,82	48,50	67,10
V18_30	50,62	6,99	47,0	75,0	V18_40	50,58	6,93	47,0	75,0
V18_31	54,39	4,94	46,0	71,0	V18_41	53,5	1,36	53,0	60,0
V18_32	57,87	6,63	48,0	72,0	V18_42	54,28	9,41	48,0	83,0
V18_33	47,86	1,70	47,0	55,0	V18_43	53,85	3,88	49,0	75,0
V18_34	50,23	1,62	47,0	60,0	V18_44	51,24	3,37	46,0	73,0
V18_35	52,07	6,85	44,0	76,0	V18_45	50,33	5,09	48,0	71,0
V18_36	58,66	4,36	49,0	78,0	V18_46	52,54	7,17	50,0	82,0
V18_37	58,44	7,59	53,0	82,0	V18_47	51,54	2,92	47,0	71,0
V18_38	53,42	3,65	44,0	62,0	V18_48	55,98	1,87	52,0	59,0
V18_39	51,15	3,80	47,0	65,0	V18_49	53,98	3,52	53,0	76,0
Média	53,47	4,81	47,20	69,60	Média	52,78	4,55	49,30	72,50
V18_50	57,14	4,85	53,0	80,0	Média geral	53,48	4,21	48,26	72,16
V18_51	54,47	5,62	46,0	86,0					
V18_52	48,11	2,38	46,0	62,0					
V18_53	55,63	6,31	52,0	86,0					
V18_54	53,93	5,25	49,0	87,0					
V18_55	55,61	9,29	48,0	96,0					
V18_56	54,87	6,07	48,0	77,0					
V18_57	55,99	7,90	49,0	110,0					
V18_58	55,36	7,07	44,0	103,0					
V18_59	53,37	0,92	53,0	59,0					
Média	54,44	5,56	48,80	84,60					

Tab. D.22: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhança Totalística com três atributos diferentes e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	53,14	1,61	51,0	60,0	V18_20	52,84	0,83	51,0	56,0
V18_11	56,95	0,74	55,0	60,0	V18_21	53,48	1,41	51,0	59,0
V18_12	53,97	2,03	53,0	65,0	V18_22	53,96	2,24	53,0	65,0
V18_13	52,88	0,70	51,0	56,0	V18_23	54,86	0,72	53,0	58,0
V18_14	52,88	0,70	51,0	56,0	V18_24	54,88	0,70	53,0	58,0
V18_15	53,83	2,25	53,0	62,0	V18_25	53,16	1,25	51,0	59,0
V18_16	57,42	3,83	53,0	65,0	V18_26	52,92	0,70	51,0	56,0
V18_17	46,7	2,90	44,0	60,0	V18_27	52,88	0,70	51,0	56,0
V18_18	52,96	0,77	51,0	56,0	V18_28	52,93	0,97	51,0	57,0
V18_19	52,99	0,79	51,0	56,0	V18_29	55,18	4,01	47,0	68,0
Média	53,37	1,63	51,30	59,60	Média	53,70	1,35	51,20	59,20
V18_30	52,89	7,36	47,0	69,0	V18_40	52,53	7,36	47,0	69,0
V18_31	46,7	2,90	44,0	60,0	V18_41	58,2	0,93	57,0	63,0
V18_32	54,26	3,70	51,0	62,0	V18_42	49,15	5,55	40,0	66,0
V18_33	49,7	2,06	47,0	56,0	V18_43	54,59	2,59	53,0	60,0
V18_34	51,83	2,31	47,0	60,0	V18_44	48,4	3,61	46,0	60,0
V18_35	49,13	0,83	48,0	53,0	V18_45	50,96	6,56	46,0	75,0
V18_36	60,91	5,85	56,0	79,0	V18_46	50,88	2,00	49,0	53,0
V18_37	53,07	1,51	51,0	60,0	V18_47	53,71	1,99	51,0	65,0
V18_38	46,09	3,42	44,0	56,0	V18_48	58,03	1,67	53,0	71,0
V18_39	53,65	1,41	53,0	60,0	V18_49	55,53	2,99	53,0	63,0
Média	51,82	3,13	48,80	61,50	Média	53,19	3,52	49,50	64,50
V18_50	57,95	4,47	54,0	85,0	Média geral	53,05	2,55	50,06	61,54
V18_51	55,01	0,82	53,0	58,0					
V18_52	45,63	3,62	41,0	60,0					
V18_53	55,48	3,85	52,0	63,0					
V18_54	54,39	3,27	53,0	65,0					
V18_55	52,06	6,79	45,0	66,0					
V18_56	50,0	0,0	50,0	50,0					
V18_57	56,37	3,41	49,0	61,0					
V18_58	50,93	4,49	44,0	62,0					
V18_59	54,26	0,96	54,0	59,0					
Média	53,20	3,16	49,50	62,90					

Tab. D.23: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Totalística com três atributos diferentes e o modo de operação paralelo de AC.

D.2.4 Vizinhaça Totalística e o atributo nível dinâmico

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	51,41	1,95	44,0	59,0	V18_20	52,52	1,05	51,0	56,0
V18_11	55,17	0,59	55,0	58,0	V18_21	55,11	2,91	51,0	60,0
V18_12	51,94	1,41	51,0	58,0	V18_22	51,18	0,76	51,0	57,0
V18_13	51,16	0,60	51,0	54,0	V18_23	53,97	1,31	53,0	58,0
V18_14	51,16	0,60	51,0	54,0	V18_24	53,19	0,66	53,0	56,0
V18_15	51,16	0,60	51,0	54,0	V18_25	51,09	0,45	51,0	54,0
V18_16	51,16	0,60	51,0	54,0	V18_26	51,19	0,66	51,0	54,0
V18_17	46,08	3,79	44,0	56,0	V18_27	51,12	0,48	51,0	53,0
V18_18	51,78	3,11	51,0	71,0	V18_28	51,14	0,57	51,0	54,0
V18_19	51,16	0,60	51,0	54,0	V18_29	57,88	1,67	55,0	60,0
Média	51,21	1,38	50,0	57,20	Média	52,83	1,05	51,80	56,20
V18_30	48,12	4,65	47,0	69,0	V18_40	47,9	4,24	47,0	69,0
V18_31	46,23	3,83	44,0	56,0	V18_41	57,2	0,79	57,0	62,0
V18_32	58,35	10,43	51,0	77,0	V18_42	49,82	7,82	40,0	73,0
V18_33	51,8	1,32	51,0	56,0	V18_43	53,43	1,08	53,0	57,0
V18_34	47,69	1,55	47,0	52,0	V18_44	53,28	3,35	48,0	60,0
V18_35	49,03	0,17	49,0	50,0	V18_45	46,78	1,95	46,0	58,0
V18_36	57,23	2,82	56,0	79,0	V18_46	54,7	0,89	53,0	58,0
V18_37	54,14	4,43	51,0	73,0	V18_47	56,61	4,71	51,0	76,0
V18_38	52,03	3,84	44,0	56,0	V18_48	61,09	6,41	49,0	79,0
V18_39	56,13	4,41	55,0	80,0	V18_49	59,22	5,75	57,0	84,0
Média	52,07	3,74	49,50	64,80	Média	54,00	3,69	50,10	67,60
V18_50	60,44	8,61	54,0	92,0	Média geral	53,02	2,82	50,14	62,80
V18_51	57,84	2,86	53,0	63,0					
V18_52	43,99	3,58	41,0	56,0					
V18_53	66,85	5,06	54,0	83,0					
V18_54	50,76	6,89	47,0	74,0					
V18_55	56,36	6,67	48,0	76,0					
V18_56	50,78	0,94	50,0	55,0					
V18_57	49,28	1,33	49,0	58,0					
V18_58	55,3	3,33	45,0	61,0					
V18_59	58,54	3,41	52,0	64,0					
Média	55,01	4,26	49,30	68,20					

Tab. D.24: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Simples do Gauss18, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	51,47	1,72	44,0	56,0	V18_20	51,37	0,86	51,0	56,0
V18_11	55,03	2,39	49,0	63,0	V18_21	51,42	1,46	49,0	58,0
V18_12	51,63	1,69	44,0	56,0	V18_22	51,31	2,14	44,0	59,0
V18_13	51,05	2,11	47,0	59,0	V18_23	50,63	2,40	49,0	56,0
V18_14	51,05	2,19	47,0	59,0	V18_24	52,89	2,49	47,0	59,0
V18_15	51,05	2,19	47,0	59,0	V18_25	53,16	1,28	47,0	58,0
V18_16	51,05	2,19	47,0	59,0	V18_26	51,16	2,17	47,0	61,0
V18_17	44,81	1,94	44,0	54,0	V18_27	51,05	2,19	47,0	59,0
V18_18	48,63	2,62	47,0	56,0	V18_28	51,07	2,15	44,0	56,0
V18_19	51,07	2,20	47,0	59,0	V18_29	52,48	1,41	47,0	60,0
Média	50,68	2,12	46,30	58,0	Média	51,65	1,85	47,20	58,20
V18_30	47,16	0,60	47,0	50,0	V18_40	47,19	0,66	47,0	50,0
V18_31	45,19	2,58	44,0	56,0	V18_41	57,11	0,67	53,0	59,0
V18_32	52,72	2,71	48,0	62,0	V18_42	48,73	2,06	40,0	54,0
V18_33	49,68	2,47	46,0	59,0	V18_43	53,48	1,08	53,0	56,0
V18_34	47,5	1,09	47,0	52,0	V18_44	46,04	0,28	46,0	48,0
V18_35	48,54	1,47	44,0	59,0	V18_45	49,05	3,60	46,0	57,0
V18_36	56,26	0,80	53,0	59,0	V18_46	53,48	0,50	53,0	54,0
V18_37	56,14	5,29	49,0	75,0	V18_47	51,03	1,71	49,0	56,0
V18_38	53,21	3,61	47,0	61,0	V18_48	55,19	3,03	52,0	60,0
V18_39	54,18	2,81	47,0	70,0	V18_49	54,12	3,12	53,0	74,0
Média	51,05	2,34	47,20	60,30	Média	51,54	1,67	49,20	56,80
V18_50	57,93	1,43	52,0	69,0	Média geral	51,49	2,11	47,80	59,32
V18_51	54,36	1,35	53,0	61,0					
V18_52	46,1	4,42	41,0	56,0					
V18_53	53,22	3,21	52,0	63,0					
V18_54	51,15	4,49	47,0	77,0					
V18_55	58,52	2,88	52,0	70,0					
V18_56	50,65	1,16	50,0	61,0					
V18_57	49,89	1,85	48,0	59,0					
V18_58	49,25	4,15	44,0	56,0					
V18_59	54,29	1,23	52,0	61,0					
Média	52,53	2,61	49,10	63,30					

Tab. D.25: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Evolução Conjunta do Gauss18 + 5 variações, Vizinhaça Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.

Grafo	T_{med}	T_{dp}	T_{min}	T_{max}	Grafo	T_{med}	T_{dp}	T_{min}	T_{max}
V18_10	51,0	0,0	51,0	51,0	V18_20	51,0	0,0	51,0	51,0
V18_11	55,03	0,30	55,0	58,0	V18_21	56,51	5,59	49,0	74,0
V18_12	51,19	0,70	51,0	56,0	V18_22	51,06	0,60	51,0	57,0
V18_13	51,0	0,0	51,0	51,0	V18_23	53,03	0,30	53,0	56,0
V18_14	51,0	0,0	51,0	51,0	V18_24	53,0	0,0	53,0	53,0
V18_15	51,0	0,0	51,0	51,0	V18_25	59,4	2,23	54,0	66,0
V18_16	51,0	0,0	51,0	51,0	V18_26	51,0	0,0	51,0	51,0
V18_17	44,0	0,0	44,0	44,0	V18_27	51,0	0,0	51,0	51,0
V18_18	51,0	0,0	51,0	51,0	V18_28	51,0	0,0	51,0	51,0
V18_19	51,0	0,0	51,0	51,0	V18_29	59,39	2,98	55,0	68,0
Média	50,72	0,10	50,70	51,50	Média	53,63	1,17	51,90	57,80
V18_30	47,63	1,56	47,0	57,0	V18_40	47,55	1,68	47,0	57,0
V18_31	44,0	0,0	44,0	44,0	V18_41	57,0	0,0	57,0	57,0
V18_32	51,0	0,0	51,0	51,0	V18_42	53,53	5,79	49,0	70,0
V18_33	52,46	1,21	51,0	55,0	V18_43	53,25	0,92	53,0	57,0
V18_34	47,0	0,0	47,0	47,0	V18_44	48,28	1,47	48,0	58,0
V18_35	48,85	0,97	48,0	57,0	V18_45	46,09	0,90	46,0	55,0
V18_36	56,0	0,0	56,0	56,0	V18_46	54,96	1,04	54,0	59,0
V18_37	51,0	0,0	51,0	51,0	V18_47	55,93	3,99	51,0	64,0
V18_38	52,96	2,12	51,0	57,0	V18_48	57,0	0,0	57,0	57,0
V18_39	56,76	0,74	55,0	58,0	V18_49	58,03	2,10	53,0	60,0
Média	50,76	0,66	50,10	53,30	Média	53,16	1,78	51,50	59,40
V18_50	54,11	0,54	54,0	57,0	Média geral	52,52	1,18	51,24	56,84
V18_51	54,64	4,73	53,0	76,0					
V18_52	53,26	5,35	47,0	66,0					
V18_53	52,0	0,0	52,0	52,0					
V18_54	53,43	1,13	53,0	57,0					
V18_55	60,14	2,71	52,0	72,0					
V18_56	51,8	2,42	50,0	59,0					
V18_57	50,05	0,60	49,0	56,0					
V18_58	58,06	3,85	54,0	62,0					
V18_59	56,15	0,99	56,0	65,0					
Média	54,36	2,23	52,0	62,20					

Tab. D.26: Valores obtidos nas 50 variações do Gauss18: aplicação das regras descobertas com a Coevolução do Gauss18 + 9 variações aleatórias, Vizinhança Totalística com atributo nível dinâmico e o modo de operação paralelo de AC.